НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина»

Виконав:
студент IV курсу, групи КП-61
Півненко Олексій Володимирович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.,
Люшенко Леся Анатоліївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.ф.-м.н., доцент,
Шубенкова Ірина Анатоліївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

«ЗАТВЕРДЖУЮ»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Півненко Олексію Володимировичу

1. Тема проєкту «Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина», керівник проєкту Люшенко Леся Анатоліївна, к.т.н., старший викладач, наказом по університету від «25» травня 2020 р. №1181-с.

2. Термін подання студентом проєкту «12» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень поставленої задачі;
   - обґрунтування вибору засобів реалізації;
   - опис розробленого веб-додатку;
   - аналіз розробленого веб-додатку.

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних(креслення);
   - алгоритм редагування інтерфейсу(креслення);
   - діаграма прецедентів (плакат);
   - структурна схема системи (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.11.2019	
2.	Розроблення та узгодження технічного завдання	27.11.2019	
3.	Розроблення структури веб-додатку	16.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	27.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	22.02.2020	
7.	Програмна реалізація веб-додатку	11.03.2020	
8.	Тестування веб-додатку	18.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	29.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	14.04.2020	
11.	Підготовка графічної частини дипломного проєкту	22.04.2020	
12.	Оформлення документації дипломного проєкту	28.05.2020	

Студент Олексій ПІВНЕНКО

Керівник проєкту Леся ЛЮШЕНКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню веб-сайту кафедри програмного забезпечення комп’ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

У роботі виконано порівняльний аналіз існуючих веб-додатків вищих учбових закладів, проаналізовано нормативні документи що регламентують роботу веб-сайтів підрозділів університету, обґрунтовано вибір технологій та допоміжних бібліотек серверної частини для реалізації даного веб-додатку. Розроблений веб-додаток надає користувачам структуровану по розділам інформацію про діяльність кафедри програмного забезпечення комп’ютерних систем, а також всю необхідну інформацію що стосується навчання на кафедрі. Також сервіс надає адміністраторам можливість доповнення та редагування контенту.

В даному проєкті розроблено та досліджено: архітектуру серверної частини веб-додатку, алгоритм авторизації користувача, структуру бази даних, стрічку новин RSS.

ABSTRACT

This diploma project is devoted to the development of website of Computer Systems Software Department of the Faculty of Applied Mathematics of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

During the writing of the diploma project was carried out a comparative analysis of the existing web applications of higher education institutions, analyzes the regulations governing the work of websites of university departments, substantiates the choice of technologies and auxiliary libraries of the server part for the implementation of this web application The developed web application provides users with structured information on the activities of the Department of Computer Systems Software, as well as all the necessary information regarding studying at the department. The service also gives administrators the ability to add and edit content.

In this project the following is developed and researched: architecture of the server part of the web application, the authorization algorithm, database structure, RSS feed.

ДП.045440-01-90 Веб-сайт кафедри програмного забезпечення комп’ютерних систем. Серверна частина. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Веб-сайт кафедри	5	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Технічне завдання		
ДП.045440-03-81	Веб-сайт кафедри	51	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-сайт кафедри	4	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-сайт кафедри	9	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Керівництво		
	адміністратора		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Веб-сайт кафедри	1	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Алгоритм редагування		
	інтерфейсу. Схема		
	роботи алгоритму.		
ДП.045440-07-99	Веб-сайт кафедри	1	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Структура бази даних.		
	ERD-діаграма		
ДП.045440-08-98	Веб-сайт кафедри	1	
	програмного		
	забезпечення		
	комп’ютерних систем.		
	Серверна частина.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2019 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сайт кафедри програмного забезпечення комп’ютерних систем. Серверна частина.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розроблений веб-сайт допоможе в вирішенні освітніх і науково-інноваційних задач кафедри з використанням сучасних інформаційних технологій, створити цілісний позитивний імідж кафедри, як підрозділу університету, обмінюватися інформацією між підрозділами університету та оперативно інформувати викладачів, співробітників та студентів про різні аспекти життя та діяльності кафедри.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинно відповідати наступним вимогам:

1) забезпечення безперервний доступ користувачів;
2) обмеження доступу до приватних ресурсів на сервері;
3) коректна обробка будь-яких запитів користувача;
4) забезпечення можливості редагування інформації;
5) використання захищеного протоколу передачі даних;
6) наявність працюючого RSS.

Вимоги до бази даних:

1) захист доступу;
2) резервне копіювання даних;
3) відмовостійкість;
4) масштабованість.

Медіа контент (відео, зображення) представлений на сайті повинен бути високої якості.

Захист інформації від несанкціонованого доступу повинен забезпечуватися за рахунок механізму авторизації. Можливість створення, редагування та видалення контенту повинна бути доступна тільки авторизованим користувачам у відповідності до їх рівня доступу.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво адміністратора;
4) креслення:
   - «Структура бази даних»;
   - «Діаграма прецедентів».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи..……....14.11.2019

Розроблення та узгодження технічного завдання…….28.11.2019

Розроблення структури програмного забезпечення……..…15.12.2019

Розроблення дизайну сторінок та графічних елементів……...03.02.2020

Програмна реалізація поставленого завдання……...17.03.2020

Тестування програмного забезпечення…….....03.04.2020

Підготовка матеріалів текстової частини проєкту……..……..28.04.2020

Підготовка матеріалів графічної частини проєкту…….....12.05.2020

Оформлення технічної документації проєкту……25.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

MVC – Model-View-Controller, архітектурний шаблон що використовується під час проєктування та розробки програмного забезпечення.

RSS – Rich Site Summary, сімейство XML-форматів, що слугує для постачання та публікації інформації, що часто змінюється.

HTTPS – Hyper Text Transfer Protocol Secure, протокол HTTP з додатковим шаром шифрування.

CMS – Content Management System, система управління контентом веб-сайту.

БД – база даних.

СКБД – система керування базами даних.

CSS – Cascading Style Sheets, каскадні таблиці стилів.

ECMAScript – стандарт мови програмування, затверджений міжнародною організацією ECMA.

HTML – Hyper Text Markup Language, стандартизована мова розмітки документів в мережі.

JVM – Java Virtual Machine, віртуальна машина, що виконує скомпільований байткод.

ACID – Atomicity, Consistency, Isolation, Durability, набір властивостей, що гарантують надійну роботу транзакцій баз даних.

SQL – мова структурованих запитів, що застосовується для управління даними в реляційних базах даних.

UNIX – сімейство переносних, багатозадачних і багатокористувацьких операційних систем.

Авторизація – керування рівнями та засобами доступу до ресурсу залежно від ідентифікатора і пароля користувача.

Автентифікація – процедура встановлення належності користувачеві певної інформації за допомогою наданого ним ідентифікатора.

HTTP – Hyper Text Transfer Protocol, прикладний протокол передачі даних у комп'ютерних мережах.

REST – Representational State Transfer, підхід до архітектури мережевих протоколів що забезпечують доступ до інформаційних ресурсів.

XML – стандарт побудови мов розмітки для даних що мають ієрархічну структуру.

JSON – текстовий формат обміну даними між комп'ютерами, що використовується переважно для передачі даних через мережу.

ВСТУП

З розвитком інтернету змінилося відношення до інформації. Майже в кожного з нас є смартфон, і у будь-який момент можемо знайти у мережі те що нас цікавить. Змінився й сам підхід до інформації, більше немає потреби зберігати інформацію на індивідуальних носіях, значно важливіше знати де можна знайти потрібну актуальну інформацію в інтернет мережі.

Зараз майже кожна організація має власний веб-сайт який представляє її діяльність в мережі, і якщо веб-сайту немає це наштовхує на сумнівні думки і знижує довіру до організації. Тобто на сьогоднішній день, веб-сайт – це по суті обличчя компанії, поглянувши на який люди зазичай складають перше враження про неї. І дуже важливо, щоб це враження було позитивним. Також важливо, щоб сайт містив необхідні функціональні можливості та контент заради якого люди його відвідують. Якщо користувача щось не влаштовує у веб-сайті, то це може викликати відчуття розчарування чи роздратування, через що ці відчуття починають асоціюватись з організацією, і зменшують лояльність до неї.

Згідно різних статистичних опитувань близько 50% людей вважають дизайн веб-сайту головним фактором при прийнятті рішення про довіру до організації. Близько 30% опитаних перестануть вивчати веб-сайт, якщо його контент буде непривабливим. Також 50% користувачів очікують що швидкість завантаження сторінки буде менше двох секунд, а 35% вказали що перестануть взаємодіяти з веб-сайтом, якщо сторінка завантажується більше шести секунд. Приблизно дві третини користувачів переглядають веб-сайти з мобільних пристроїв. Близько 30% сприймають відсутність адаптивного дизайну як відсутність турботи про користувачів, що знижує їх довіру до установи. Приблизно 35% вказали що виберуть інший сайт у пошуку, якщо той на який вони перейшли не адаптований для перегляду на мобільних пристроях.

Основною метою дипломного проєкту є створення нового веб-сайту кафедри програмного забезпечення комп’ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1. Аналіз предметної області

Функціонування сайтів наукових підрозділів Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» регламентується статутом університету, чинним законодавством, наказами та розпорядженнями ректора, положенням про інформаційно-освітній портал НТУУ «КПІ», внутрішніми розпорядженнями структурного підрозділу університету, розпорядженнями проректора (перспективний розвиток).

Згідно нормативних документів основними завданнями сайту кафедри є:

- обмін інформацією між різними підрозділами університету;
- вирішення науково-інноваційних та освітніх задач кафедри й університету за допомогою використання сучасних інформаційних технологій;
- оперативне та об'єктивне інформування викладачів, студентів, аспірантів, співробітників, докторантів, випускників, абітурієнтів, ділових партнерів, а також інших зацікавлених осіб про різні аспекти діяльності кафедри;
- створення позитивного іміджу кафедри, як університетського підрозділу, що здатен конкурувати на міжнародному ринку послуг у сфері науки та освіти.

Чинна версія сайту відповідає вимогам описаним у нормативних документах не в повному обсязі. Також дизайн та технічні характеристики веб-додатку не виправдовують очікування користувачів, що звикли користуватися веб-додатками виконаним з урахуванням сучасних тенденцій у сфері веб-розробки.

Метою даного дипломного проєкту є розробка серверної частини веб-сайту кафедри програмного забезпечення комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», яка покликана виправити існуючі недоліки та привести роботу веб-сайту у відповідність до вищезазначених нормативних документів.

1.2. Аналіз аналогічних застосунків

З метою аналізу аналогів були розглянуті веб-додатки різних університетів України та світу. Були визначені їх переваги та недоліки.

1.2.1. Веб-додаток Ягеллонського університету у Кракові

Структура веб-додатку [1] залежить від вибраної мови інтерфейсу. Для англійської локалізації доступні наступні розділи:

- про університет;
- навчання;
- кадровий склад;
- наукова діяльність;
- міжнародна співпраця.

Також на головній сторінці веб-додатку зображеної на рис. 1, наявні посилання на сторінки університету в соціальних мережах. Якість медіа контенту представленого на сторінках веб-додатку висока.

До переваг даного рішення можна віднести сучасний дизайн, використання захищеного протоколу передачі даних, можливість зміни розміру шрифту, можливість переключення до більш контрастної версії інтерфейсу, локалізацію інтерфейсу для декількох мов, наявність особистого кабінету користувача, висока швидкодія, підтримка мобільних пристроїв, наявність стрічки новин RSS, тощо.

Серверна частина розроблена з використанням мови програмування Java. Клієнтська частина виконана з допомогою HTML, CSS, різноманітних JavaScript бібліотек. До веб-додатку підключений модуль аналітики Google Analytics.

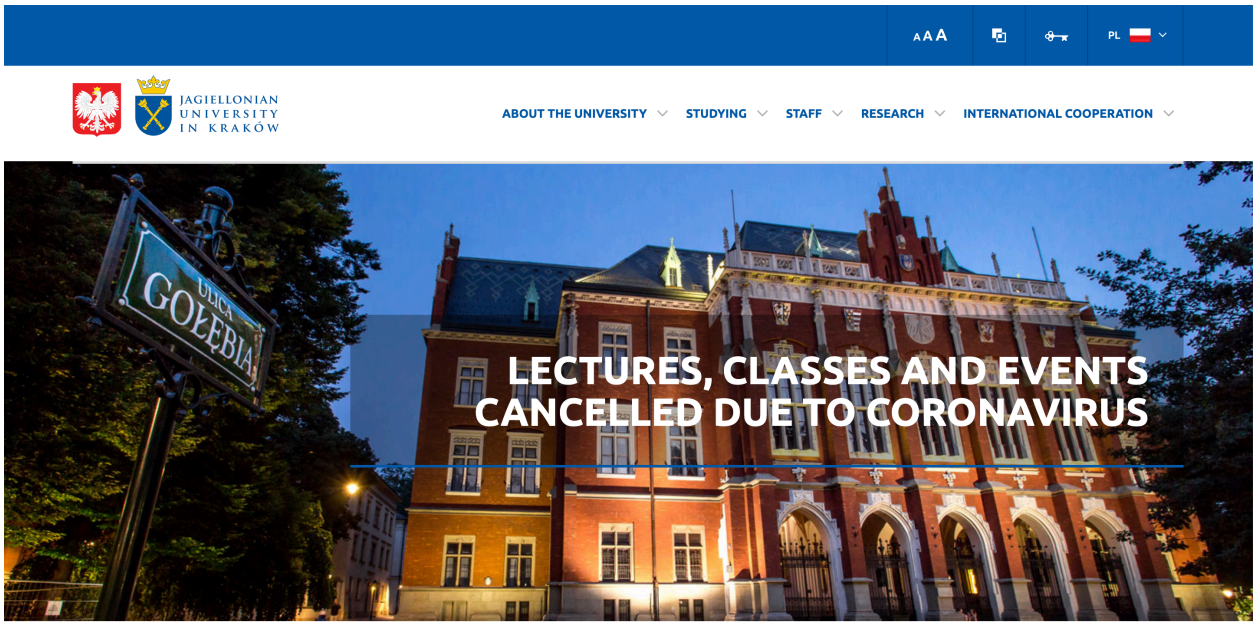


Рис. 1. Головна сторінка веб-додатку Ягеллонського університету

1.2.2. Веб-додаток Мюнхенського технічного університету

Структурно веб-додаток [2] поділяється на наступні розділи:

- про університет;
- наука;
- інновації;
- навчання;
- студентське життя;
- міжнародні зв’язки.

Медіа контент представлений на сторінках веб-додатку у високій якості. Також на головній сторінці, зображеної на рис. 2, містяться посилання на сторінки університету в соціальних мережах.

Серед переваг можна виділити мінімалістичний сучасний дизайн, відсутність зайвих елементів інтерфейсу, зручну навігацію по сайту, використання захищеного протоколу передачі даних, високу швидкодію, стрічку новин RSS.

До недоліків можна віднести відсутність особистого кабінету користувача.

Серверна частина розроблена з використанням мови програмування PHP. Для розробки клієнтської частини використовувалися HTML, CSS, jQuery.

Рис. 2. Головна сторінка веб-додатку Мюнхенського технічного університету

1.2.3. Веб-додаток Національного медичного університету ім. О.О. Богомольця

Структура веб-додатку [3] містить наступні розділи:

- загальні відомості;
- інформація для абітурієнтів;

- інформація для студентів;
- інформація для випускників;
- наука.

З переваг можна виділити наявність пошуку по сайту та наявність стрічки новин RSS.

До недоліків можна віднести застарілий дизайн, повільну швидкість завантаження сторінок, відсутність підтримки мобільних пристроїв, відображення копірайту з закінченим терміном дії, нема індивідуального favicon, деякі зображення не завантажуються взагалі .

Суперечливим є розділення доступу до інформації залежно від мови, з одного боку це дозволяє іноземним студентам економити час не переглядаючи інформацію що їх не стосується, з іншого боку сам інтерфейс не змінюється і переходи за посиланнями з обмеженим доступом викликає лише роздратування.

Хоч на головній сторінці веб-додатку, зображеній на рис. 3, наявні посилання на сторінки університету у соціальних мережах, деякі посилання працюють некоректно перенаправляючи на головну сторінку ресурсу, замість сторінки вищого навчального закладу. Крім того, деякі посилання ведуть на ресурси що припинили свою роботу або заблоковані на території України.

Серверна частина розроблена на CMS WordPress з використанням мови програмування PHP та СКБД MySQL. Клієнтська частина розроблена з використанням HTML та CSS. До веб додатку підключений модуль аналітики Google Analytics.

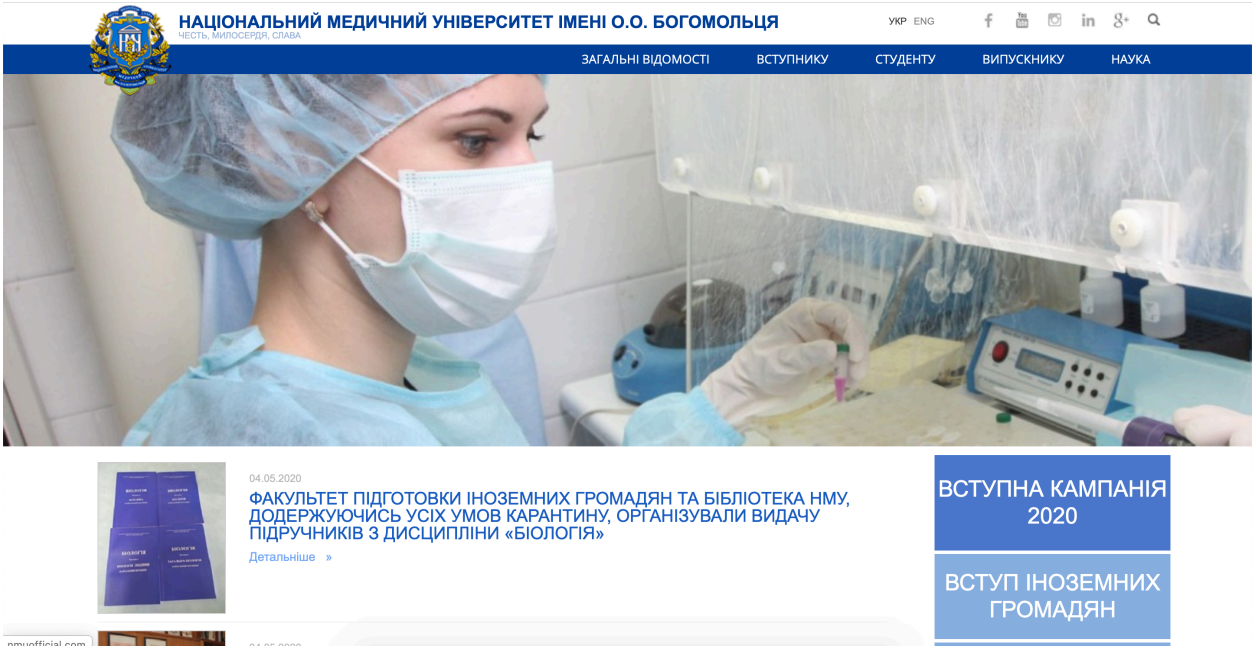


Рис. 3. Головна сторінка веб-додатку НМУ ім. О.О. Богомольця

Результати аналізу наведено у табл. 1.

Таблиця 1

Порівняльна таблиця програмних рішень

Критерії / Програмні рішення	Сучасний дизайн	Швидкодія	Підтримка мобільних пристроїв	Захищене підключення
Ягеллонський університет	+	+	+	+
Мюнхенський технічний університет	+	+	–	+
НМУ ім. Олександра Богомольця	–	–	–	+

За результатами аналізу було встановлено, що веб-додаток Ягеллонського університету є найкращим серед порівнюваних аналогів. Він має новітній дизайн, адаптивний інтерфейс зі спеціальними можливостями для людей що мають вади зору, високу швидкодію, захищене підключення, стрічку новин RSS, тощо.

В той же час веб-додаток Національного медичного університету ім. О.О. Богомольця найкраще відображає недоліки які веб-додаток містити не повинен.

1.3. Актуальність розробки веб-додатку

У підрозділі з аналізом аналогічних застосунків були наведені приклади веб-додатків вищих навчальних закладів України та Європи. Хоча деякі функціональні можливості та характеристики є спільними кожен аналог розроблений відповідно до потреб конкретної організації, яку він представляє.

Виходячи з відмінностей у структурі та наповненню сайтів жоден з аналогів не може бути застосований для вирішення поставленої задачі.

Чинна версія сайту містить ряд недоліків, а саме:

- невідповідність структури веб-сайту вимогам описаним у положенні про типовий сайт кафедри НТУУ «КПІ»;
- застарілий дизайн;
- використання незахищеного протоколу передачі даних;
- повільне завантаження сторінок.

У результаті аналізу було сформовано наступні вимоги до веб-додатку в цілому та серверної частини зокрема:

- забезпечення безперервний доступ користувачів;
- висока швидкість роботи;
- використання захищеного протоколу передачі даних;
- обмеження доступу до приватних ресурсів на сервері;

- забезпечення можливості редагування інформації;
- коректна обробка будь-яких запитів користувача;
- наявність працюючого RSS.

Вимоги до БД:

- захист доступу;
- резервне копіювання даних;
- масштабованість;
- відмовостійкість.

Структура веб-сайту повинна містити наступні розділи:

- загальна інформація про підрозділ;
- вступ;
- навчання;
- наука;
- персонал/список викладачів;
- студентське життя;
- зарубіжне партнерство;
- новини/оголошення.

Медіа контент (відео, зображення) представлений на сайті повинен бути високої якості.

Порядок забезпечення захист інформації веб-сайту визначений НД ТЗІ 2.5- 010-03 та Законом України про захист інформації в інформаційно-комунікаційних системах.

Захист інформації від несанкціонованого доступу повинен забезпечуватися за рахунок механізму авторизації. Можливість створення, редагування та видалення контенту повинна бути доступна тільки авторизованим користувачам у відповідності до їх рівня доступу.

1.4. Висновки до розділу

На сьогоднішній день більшість веб-додатків вітчизняних вищих навчальних закладів є застарілими. Враховуючи той факт, що веб-сайт є одним з головних ресурсів для пошуку інформації, люди змушені його використовувати для задоволення власних потреб, за відсутності кращого варіанту. Такі речі як застарілий дизайн, складний заплутаний інтерфейс, відсутність підтримки мобільних пристроїв, повільна швидкість роботи не виправдовують очікування людей і погіршують ставлення до установи яку даний веб-сайт представляє. Також через невідповідність веб-сайту підрозділу чітко визначеній структурі призводить до того що користувач не може знайти потрібну інформацію через її відсутність.

Розроблюваний веб-сайт кафедри програмного забезпечення комп’ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» покликаний вирішити цю проблему відповідно до сформульованих вимог.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Оскільки розроблюваний програмний продукт має бути представлений у вигляді веб-додатку вибір мови програмування формувався згідно більш вживаних мов програмування, які використовуються для розробки даного типу ПЗ.

Мова програмування має відповідати таким вимогам:

- наявність функціональних можливостей призначених для розробки веб-додатків;
- наявність великої кількості бібліотек доступних для використання;
- підтримка різних парадигм програмування.

Мова програмування Python

Python – високорівнева мова програмування загального призначення зі строгою динамічною типізацією. За рахунок виразного синтаксису і легкості освоєння набула широкої популярності. Основними сферами застосування є аналіз даних, розробка веб додатків, прототипування різних систем, скрапінг веб-ресурсів, написання скриптів для взаємодії з операційною системою, тощо.

Основними архітектурними особливостями є:

- динамічна типізація;
- автоматичне керування пам'яттю;
- повна інтроспекція;
- наявність механізму обробки виключень;
- підтримка паралельних обчислень у різних потоках;
- наявність високорівневих структури даних.

Наявність модульної системи та пакетів модулів сприяє кращій організації та перевикористанню кодової бази. Python належить до мультипарадигмальних мов програмування.

До переваг Python можна віднести:

- низький поріг входу та значна швидкість створення програмного забезпечення, що зумовлені простотою синтаксису мови та великою кількістю навчальних матеріалів та документації;
- наявність великої кількості бібліотек та програмних модулів у відкритому доступі;
- легкість читання коду та його подальшої модифікації.

Головним недоліком Python є відносно низька швидкодія зумовлена використанням інтерпритатора.

Мова програмування C#

C# – мова програмування високого рівня що належить до об'єктно орієнтованих мов. Відноситься до сімейства C-подібних мов, вважається що синтаксис найбільш схожий на синтаксис Java або C ++ [5]. Першочергово розроблена для платформи NET під керуванням системи Windows, як аналог мови Java, що дозволило врахувати всі переваги та недоліки прототипу при розробці. C# має: статичну типізацію, підтримку делегатів, атрибутів та подій.

До переваг C# можна віднести:

- велика кількість бібліотек доступних через пакетній менеджер;
- C-подібний код зрозумілий для людей з досвідом розробки на мовах даного сімейства;
- наявність технологій що дозволяють розроблювати крос-платформні застосунки;
- наявність потужної IDE.

Недоліком C# є історична прив’язка до платформи .NET, що доволі довгий час була доступна тільки під ОС Windows. На даний момент компанія створила інструменти для розробки крос-платформних додатків, однак під інші системи все ще немає зручних інструментів для розробки програмного забезпечення з використанням даної мови. До того ж для якісної розробки бажано застосовувати такі комерційні інструменти, як наприклад ReSharper .

Мова програмування Javascript

Мова програмування Javascript [6] є реалізацією стандарту ECMAScript. Належить до мов з динамічною слабкою типізацією. Першочергово розроблялася для створення динамічного контенту в браузерах користувачів. За допомогою середовища Node.js стало можливим використання даної мови на стороні сервера, що дозволило спеціалістам розробляти повноцінні веб-додатки з застосуванням однієї мови програмування. JavaScript підтримує різні парадигми програмування.

До переваг JavaScript належать:

- надзвичайно стрімкий розвиток;
- можливість вставляти JavaScript код в HTML розмітку;
- можливість виконання коду не тільки в браузері, а й на сервері;
- широкий вибір готових бібліотек.

Головним недоліком використання мови програмування JavaScript є динамічна типізація, що є перевагою при прототипуванні, однак суттєвим недоліком, що заважає тестуванню при розробці великих високонавантажених систем.

Мова програмування Java

Java [7] – сильно типізована об'єктно-орієнтована мова програмування, розроблена під назвою “Oak” у відділі компанії Sun Microsystems . На даний

власником є компанія Oracle яка і займається розвитком мови. Поточною версією є Java 14, а актуальною версіє з довгим терміном підтримки є Java 11.

У час створення набула широкої популярності за рахунок проривної концепції віртуальної машини що дозволила досягти крос-платформності. Java код компілюється у байт-код і потім може використовуватися на будь-якій платформі, для якої реалізовано віртуальну машину. В минулому часто використовувалася для написання програм під різні мобільні пристрої та побутову техніку, зараз застосовується переважно в корпоративному сегменті для створення й підтримки високонавантажених надійних систем. Хоча Java є мультипарадигмальною мовою програмування деякі парадигми відрізняються від їх загальноприйнятого вигляду і тому не набули широкого застосування.

До переваг Java належать:

- стабільність роботи та наявність перевірених часом рішень;
- повна зворотна сумісність розроблених додатків;
- лише незначні зміни синтаксису на протязі існування мови.

До недоліків Java часто відносять «багатослівний» синтаксис, а також недостатню продуктивність без додаткових налаштувань віртуальної машини, та довгу компіляцію коду, що сповільнює розробку.

Обрана мова програмування

В результаті аналізу вищепрелічених мов були визначені критичні недоліки що змушують відмовитись від їх використання для розроблення даної системи:

- розробка на мові C# вимагає використання програмного забезпечення з комерційною ліцензією, що не є доцільним для розробки даного дипломного проєкту.
- Python хоч і сприяє швидкому створенню ПЗ, але лише за умови, що розробники володіють значим досвідом програмування з

використанням цієї мови, зважаючи що розробники даного продукту цього досвіду не мають, вищезгадана перевага повністю нівелюється, враховуючи це, а також низьку швидкодію Python є не найкращим вибором для розробки;

- мова програмування Java не є найкращим варіантом через порівняно надмірний синтаксис а також повільну швидкість роботи.

Отже, враховуючи попередній аналіз та вимоги до веб-додатку у якості мови програмування для розроблюваного ПЗ був вибраний Javascript. Дана мова має простий та зрозумілий синтаксис, дуже швидко розвивається, має широкий вибір бібліотек та фреймворків, може використовуватись для розробки як клієнтської так і серверної частин.

2.2. Обгрунтування вибору бази даних

2.2.1. *Загальний огляд баз даних та вибір моделі*

Усі бази даних, що існують на даний момент поділяються на два типи: реляційні, або як їх ще називають SQL, та нереляційні, або NoSQL. Різниця між ними полягає в тому яку модель даних вони представляють: реляційну або нереляційну. Реляційні бази представляють дані у вигляді таблиць, на відміну від нереляційних, де спосіб представлення даних залежить від типу самої бази.

Нереляційні бази поділяються на:

- стовпчикові (колонкові);
- документні;
- ключ-значення;
- графові;
- об’єктні.

Реляційні бази даних зберігають дані, структура яких чітко визначена на момент створення бази. Для взаємодії з базою даних використовується спеціальна мова запитів SQL. Оператори даної мови дозволяють проводити

внесення нових даних до бази даних, їх оновлення та видалення, а також читання. Для з’єднання таблиць бази між собою використовують зовнішні ключі, що також вирішує проблему неконсистентності даних. Незважаючи на те, що всі СУБД використовують SQL для доступу до бази, більшість з них мають Тим не менш, стандартні можливості SQL дозволяють виконувати всі необхідні операції, просто без додаткової зручності.

Переваги реляційних баз даних [8]:

- чітко виражена структура даних;
- підтримка стандартів SQL;
- надійність – історично реляційні бази старіші за нереляційні, що означає що більшість типових проблем для них вже вирішено;
- популярність – на даний момент за різними оцінками більше 70% всіх баз даних що застосовуються у світі є реляційними, що зумовлює велику кількість висококваліфікованих фахівців з досвідом роботи з реляційними БД, а також велику кількість документації
- безпека – реляційні бази забезпечують підтримку користувачів різних типів та розмежування доступу між ними в залежності від наданих ним прав;
- підтримка механізму транзакцій – групи логічних операцій що можуть виконуватись як одна, а надійність забезпечується принципами ACID;
- можливість підвищення продуктивності роботи з даними за рахунок індексації;
- потужні вбудовані інструменти обробки даних, до яких відносяться тригери та процедури.

Недоліки реляційних баз даних:

- швидкодія – для коректного представлення відношень між сутностями даних в реляційних БД використовують процес під назвою нормалізація, що через велику кількість таблиць зумовлює потребу їх з’єднання у запитах і призводить до зниження швидкості й ускладнення самих запитів для підвищення продуктивності яких використовують зворотній процес денормалізації бази даних, що призводить до проблеми надмірності даних, що означає що при використанні реляційних БД часто виникає потреба шукати компроміс між нормалізацією та швидкодією;
- масштабування – незважаючи на можливість вертикального масштабування, організувати горизонтальне масштабування реляційної бази даних досить складно;
- складна зміна схеми бази даних – при використанні реляційних баз даних структура даних має бути визначена наперед, що означає чим складніша схема бази даних тим складніше її змінити у майбутньому;
- невідповідність реляційної моделі до певного класу задач – незважаючи на універсальність реляційної моделі є задачі для яких її використання створює більше проблем ніж, якщо була використана база даних з відповідною моделлю представлення даних, серед яких можна виділити задачу обходу графу, що при використанні реляційної моделі може бути проблемним через велику кількість з’єднань таблиць у запитах.

NoSQL бази даних дозволяють зберігати неструктуровані або напівструктуровані дані. Ці дані моделюються засобами, що відрізняються від тих які використовуються при побудові реляційних базах даних. NoSQL бази даних зазвичай використовуються для вирішення конкретного ряду проблем.

На сьогоднішній день їх популярність невпинно зростає через підвищення складності програмних систем. Незважаючи на те, що іноді NoSQL рішення можуть підтримувати SQL-подібні запити, більшість з них має власні механізми роботи з даними. Відсутність чітко визначеної структури полегшує проєктування бази та робить його більш гнучким, що робить деякі операції швидшими ніж їх SQL аналоги. Перешкодами для більшого застосування відсутність стандартизації і наявність популярних реляційних аналогів з більшою підтримкою. Відсутність механізму транзакція також не сприяє покращенню їх становища.

Переваги NoSQL [9]:

- масштабованість – NoSQL бази даних підтримують зручне горизонтальне масштабування за допомогою механізму шардингу, яке є дешевшим та доступнішим за вертикальне;
- швидке відновлення – механізм автоматичної реплікації забезпечує копіювання даних і швидкого їх відновлення до збереженого стану у випадку відмови системи незалежно від її причини.

Недоліки NoSQL:

- вузька спеціалізація – використовуються зазвичай, не як загальне рішення, а для вирішення якоїсь специфічної задачі;
- відсутність стандартизації через молодий вік NoSQL рішень, а також той факт, що найчастіше такі БД є рішеннями з відкритим кодом що постійно змінюється;
- відсутність графічного інтерфейсу користувача – через невисоку популярність більшість нереляційних БД не мають графічного клієнта;

2.2.2. *Аналіз СКБД MySQL*

MySQL – система управління реляційними базами даних з відкритим вихідним кодом, створена на основі СКБД mSQL. Нині власником, що займається підтрикою продукту є компанія Oracle. СКБД портована під усі сучасні платформи, включаючи Mac, Windows, Linux і Unix. Основна сфера застосування – розробка веб-додатків. Актуальною версією на даний момент є 8.0.17.

MySQL базується на клієнт-серверній моделі, де ядром виступає MySQL сервер який оброблює всі запити до бази даних і доступний у двох варіаціях: як окрема утиліта для використання у мережевому середовищі та як бібліотека що може бути використана в іншій програмі [10]. Також для даної СКБД існує клієнт зі зручним графічним інтерфейсом користувача.

MySQL підтримує бази даних великого обсягу, а також різні типи даних, включаючи цілі числа, числа з плаваючою крапкою, різні формати дат та бінарні дані. Також наявна підтримка строкових типів змінної та фіксованої довжини.

Для забезпечення безпеки MySQL підтримує систему шифрування паролів та інформації у базі даних.

Переваги MySQL:

- популярність – одна з найпопулярніших систем, що означає відсутність дефіциту висококваліфікованих розробників з досвідом використання даної БД, а також наявність великої кількості документації та навчальних матеріалів;
- безпека – MySQL забезпечує захист інформації за рахунок системи користувачів та розмежування доступу на основі їх наданих їм прав та ролей;
- швидкість – при розробці можливості SQL не були реалізовані у повному обсязі, що дозволило підвищити продуктивність.

Недоліки MySQL:

- обмеженість – з самого початку СКБД розроблялась з акцентом на швидкість та простоту використання, через що вона не має повної відповідності усім можливостям SQL;
- платні можливості – MySQL розповсюджується з двома ліцензіями: відкритою і комерційною, через що частина функціональних можливостей доступна тільки на платній основі;
- уповільнений розвиток – через часті перекупки продукту розвиток СКБД уповільнився, бо співтовариство більше не може швидко вносити зміни, так як курс розвитку визначає компанія власник;
- зниження швидкості при великих розмірах даних – при збільшенні кількості даних єдиним способом збільшення продуктивності є застосування індексації.

2.2.3. *Аналіз СКБД PostgreSQL*

PostgreSQL – це об'єктно-реляційна система управління базами даних створена як open-source проєкт в Каліфорнійському університеті в Берклі у 1986 році. Першочергово розроблена для платформи UNIX, і згодом портована на інші популярні платформи.. PostgreSQL є проєктом з повністю відкритим вихідним кодом доступних під ліцензією PostgreSQL. Ця СКБД вирізняється стабільністю роботи та легкістю підтримки.

Особливостями PostgreSQL є можливість додавати власні функції, розроблені на різних мовах програмування для розширення стандартних функціональних можливостей, а також можливість створення власних типів даних, індексів, тощо [11].

Переваги PostgreSQL:

- паралельні та оптимізовані запити;
- забезпечення механізмів блокування;

- підтримка індексів: btree, hash, тощо;
- розподілені таблиці;
- варіативність методів реплікації: синхронна і асинхронна реплікація, логічна і фізична, часткова, тощо;
- підтримка SSL;
- додаткові типи даних – окрім стандартних типів властивих кожній реляційній базі, а також деяких додаткових типів специфічних для PostgreSQL, можна створювати свої власні;
- наявність драйверів для більшості сучасних платформ та мов програмування;
- можливість повнотекстового пошуку;
- open-source – проєкт має повністю відкритий вихідний код та підтримується спільнотою, яка займається його підтримкою, вирішує напрям розвитку та відповідає за написання документації.

Недоліки PostgreSQL:

- популярність – історичне відставання за популярністю від MySQL, зумовлене відсутністю компаній власників які б просували продукт, призвела до порівняно меншої кількості сторонніх інструментів а також меншої кількості спеціалістів з досвідом адміністрування даної СКБД;
- знижена продуктивність через виділення додаткової пам’яті на кожне підключення до бази.

Враховуючи переваги та недоліки вищеперелічених рішень для розроблення даного дипломного проєкту було обрано СКБД PostgreSQL через ширші функціональні можливості, а також той факт, що вона є повністю відкритою.

2.3. Обгрунтування вибору фреймворку серверної частини

2.3.1. Аналіз Express.js

Express.js [12] – мінімалістичний та гнучкий фреймворк, що надає широкий набір функціональних можливостей для розробки веб-додатків. Створений на основі каркасу Sinatra, що написаний на мові програмування Ruby. Серед переваг можна виділити:

- невеликий розмір;
- високу швидкодію;
- універсальність;
- популярність;
- доступна велика к-сть модулів та бібліотек.

2.3.2. Аналіз Meteor.js

Meteor.js [13] – веб-фреймворк з відкритим вихідним кодом написаний на JavaScript з використанням Node.js. Розроблений компанією Meteor Development Group у 2011 році. Meteor дозволяє швидко створювати кросплатформні веб-додатки та додатки для мобільних пристроїв. Фреймворк використовує протокол розподілених даних та шаблон проєктування publish-subscribe щоб автоматично оновлювати дані на клієнті та уникнути необхідності писати відповідний код для синхронізації.

Враховуючи вищезазначені характеристики для розробки серверної частини був обраний фреймворк Express.js через більшу популярність, що в свою чергу означає наявність більшої кількості готових бібліотек та більшу підтримку спільноти.

2.4. Висновки до розділу

У даному розділі був проведений збір вимог до технологій розробки, а саме мови програмування, бази даних та фреймворку для розробки серверної частини з урахуванням особливостей розроблюваного ПЗ.

Було проведено аналіз найпопулярніших мов програмування для розробки веб-додатків: Python, JavaScript, Java, C#. Були визначені їх переваги та недоліки.

Було обрано мову Javascript через зручність її використання для комплексної розробки серверної і клієнтської частин, наявність великої кількості бібліотек, крос-платформність.

Був проведений аналіз SQL та NoSQL баз даних, в результаті якого було визначено що для розробки даного ПЗ краще використовувати реляційну базу даних. Після цього було проаналізовано СКБД MySQL та PostgreSQL, як найбільш популярних представників БД даного типу, визначено їх переваги та недоліки. В результаті аналізу для розробки даного веб-додатку було обрано СКБД PostgreSQL.

Було розглянуто веб-фреймворки Express.js та Meteor.js для пришвидшення розробки серверної частини. Для розробки було обрано перший через вищу популярність та підтримку.

3. РОЗРОБЛЕННЯ ВЕБ-ДОДАТКУ

3.1. Загальний опис системи

Інформація розміщена на розроблюванму веб-сайті має бути подана у зручному для сприйняття вигляді відповідно до визначеної структури. Адміністратору має бути надана можливість редагування інформації.

Веб-сайт має містити розділи для представлення наступної інформації:

- загальна інформація про кафедру українською та англійською мовами;
- положення про структурний підрозділ;
- матеріально-технічне забезпечення кафедри українською та англійською мовами;
- інформація про вступ на 1 курс українською та англійською мовами;
- інформація про вступ на 5 курс українською та англійською мовами;
- освітні програми;
- навчальний план;
- перелік тем дипломних робіт попереднього і поточного років;
- дисертації магістрів попереднього і поточного років;
- інформація про працевлаштування випускників та студентів кафедри українською та англійською мовами;
- інформація про науку/наукові школи кафедри українською та англійською мовами;
- результати проведеної практики;
- публікації викладачів кафедри;
- інформація про регулярні наукові заходи на кафедрі;
- інформація про зарубіжне партнерство кафедри українською та англійською мовами;

- новини кафедри.

Веб-додаток повинен мати працездатне RSS.

3.2. Архітектура системи

Для розроблення програмного забезпечення було обрано клієнт-серверну модель архітектури.

Клієнт-серверна архітектура – архітектурний шаблон розробки ПЗ, заснований на взаємодії клієнтів та серверів через мережу (рис. 4). Клієнти та сервери представлені у вигляді програмних модулів що функціонують незалежно один від одного. Взаємодія має наступний вигляд: клієнт робить запит до сервера, сервер оброблює його і повертає результат обробки клієнту. Сервер здатен одночасно обробляти запити від багатьох клієнтів. Зазвичай у ролі клієнта виступає веб-браузер [15].

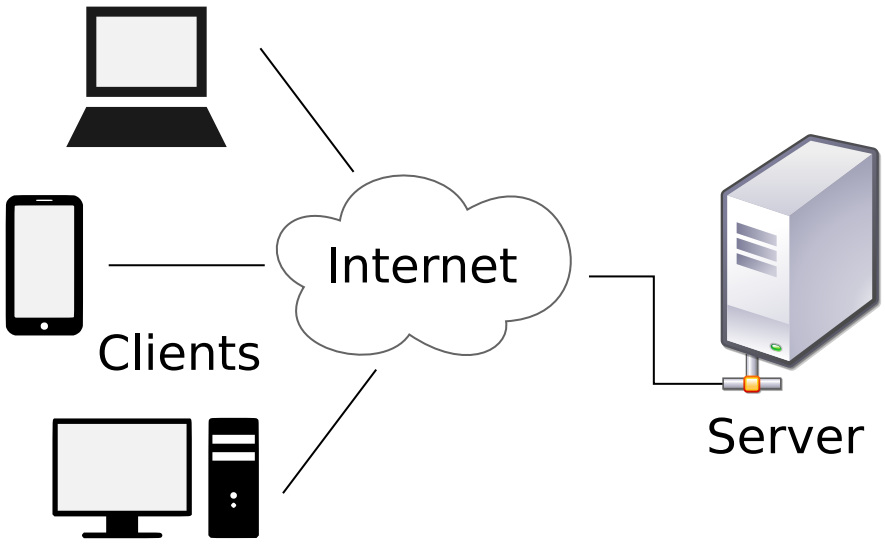


Рис. 4. Клієнт-серверна архітектура

3.3. Особливості реалізації

Реалізація серверної частини виконана на основі шаблону проєктування MVC (Model-View-Controller) з використанням фреймворку Express.js.

Схема бази даних

Для реалізації розроблюваного додатку було створено 8 взаємозв'язаних між собою таблиць:

- Staff;
- Exam_timetable;
- Consultation_timetable;
- Publications;
- Publication_items;
- Graduation_papers;
- Curriculum;
- Users.

При моделюванні бази даних було витримано баланс між нормалізацією та швидкодією. Схему бази даних зображено на рис. 5.

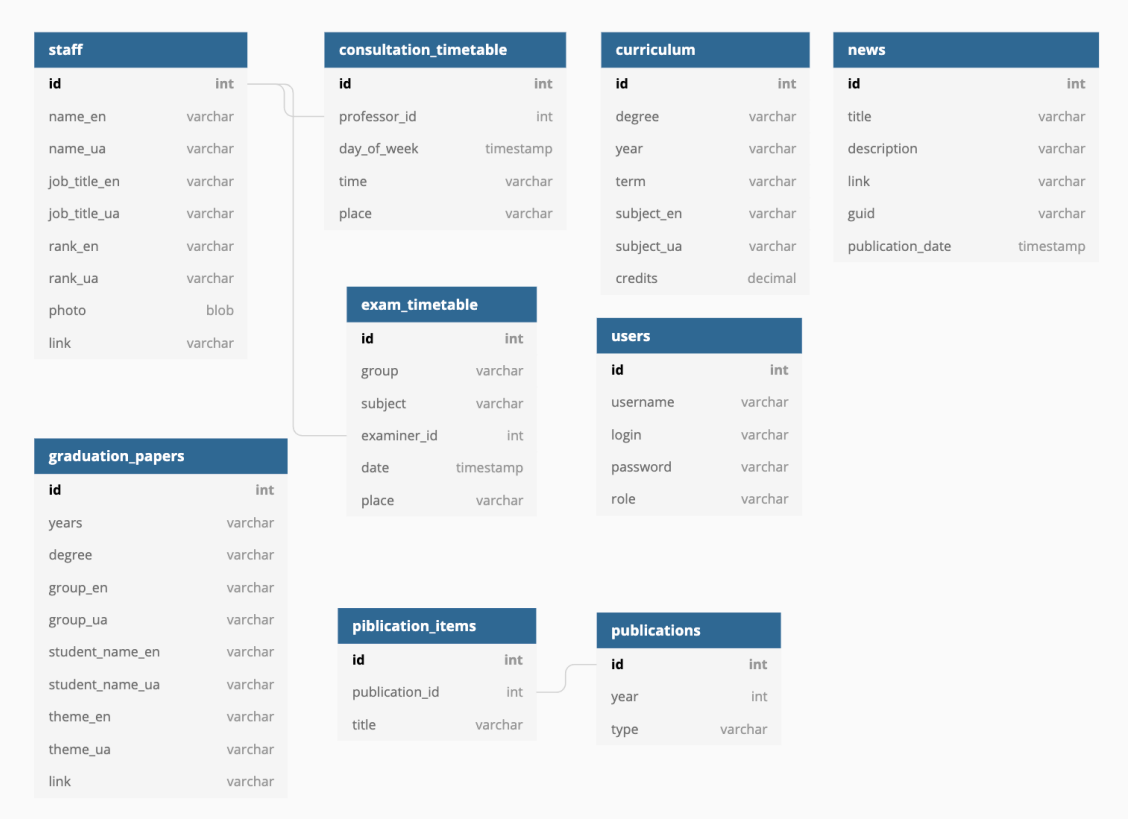


Рис. 5. Схема бази даних

Staff

Таблиця містить у собі інформацію про кадровий склад кафедри. Детальний опис назв та типів полів наведено у табл. 2.

Таблиця 2

Поля таблиці Staff

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
name_en	TEXT
name_ua	TEXT
job_title_en	TEXT
job_title_ua	TEXT
rank_en	TEXT
rank_ua	TEXT
photo	BLOB
link	TEXT

Exam_timetable

Дана таблиця представляє інформацію про розклад екзаменів під час сесії. Детальний опис назв та типів полів наведено у табл. 3.

Таблиця 3

Поля таблиці Exam_timetable

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
group	TEXT
subject	TEXT
examiner_id	INT (Foreign Key of Staff)
date	TIMESTAMP
place	TEXT

Consultation_timetable

Дана таблиця представляє інформацію про розклад консультацій викладачів. Детальний опис назв та типів полів наведено у табл. 4.

Таблиця 4

Поля таблиці Consultation_timetable

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
professor_id	INT (Foreign Key of Staff)
day_of_week	TEXT
time	TEXT
place	TEXT

Publications

Дана таблиця представляє загальну інформацію про елемент архіву публікацій. Детальний опис назв та типів полів наведено у табл. 5.

Таблиця 5

Поля таблиці Publications

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
year	INT
type	TEXT

Publication_items

Дана таблиця представляє конкретну інформацію про публікацію. Зв’язана з таблицею Publications у відношенні Many-to-One. Детальний опис назв та типів полів наведено у табл. 6.

Таблиця 6

Поля таблиці Publication_items

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
publication_id	INT (Foreign Key of Publication)
title	TEXT

Graduation_papers

Дана таблиця представляє інформацію про випускні роботи поточного і минулих років. Детальний опис назв та типів полів наведено у табл. 7.

Таблиця 7

Поля таблиці Graduation_papers

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
years	TEXT
degree	TEXT
group_en	TEXT
group_ua	TEXT
student_name_en	TEXT
student_name_ua	TEXT
theme_en	TEXT
theme_ua	TEXT
link	TEXT

Curriculum

Дана таблиця містить інформацію про учбовий план для різних наукових ступенів. Детальний опис назв та типів полів наведено у табл. 8.

Таблиця 8

Поля таблиці Curriculum

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
degree	TEXT
year	INT

Продовження табл. 8

term	INT
subject_en	TEXT
subject_ua	TEXT
credits	DECIMAL (2,2)

Users

Дана таблиця містить інформацію про авторизованих користувачів системи. Детальний опис назв та типів полів наведено у табл. 9.

Таблиця 9

Поля таблиці Users

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
username	TEXT
login	TEXT
password	TEXT
role	TEXT

Взаємодія з клієнтською частиною

Взаємодія між клієнтською та серверною частиною виконана за допомогою викликів REST API. Схема взаємодії наведена на рис. 6.

Для забезпечення взаємодії у фреймворку Express наявні спеціальні функції-обробники HTTP запитів, що називаються middleware. Вони приймають функцію яка містить аргументи двох типів:

- Request – містить всі дані про HTTP запит, що прийшов на сервер;

- Response – містить відповідь від сервера у форматі JSON, що відправляється клієнту.

Використовувати механізми REST API можуть лише авторизовані у системі користувачі. Виклик від користувачів, що не пройшли процедуру авторизації поверне відповідь з HTTP статусом 401 (Not Authorized).

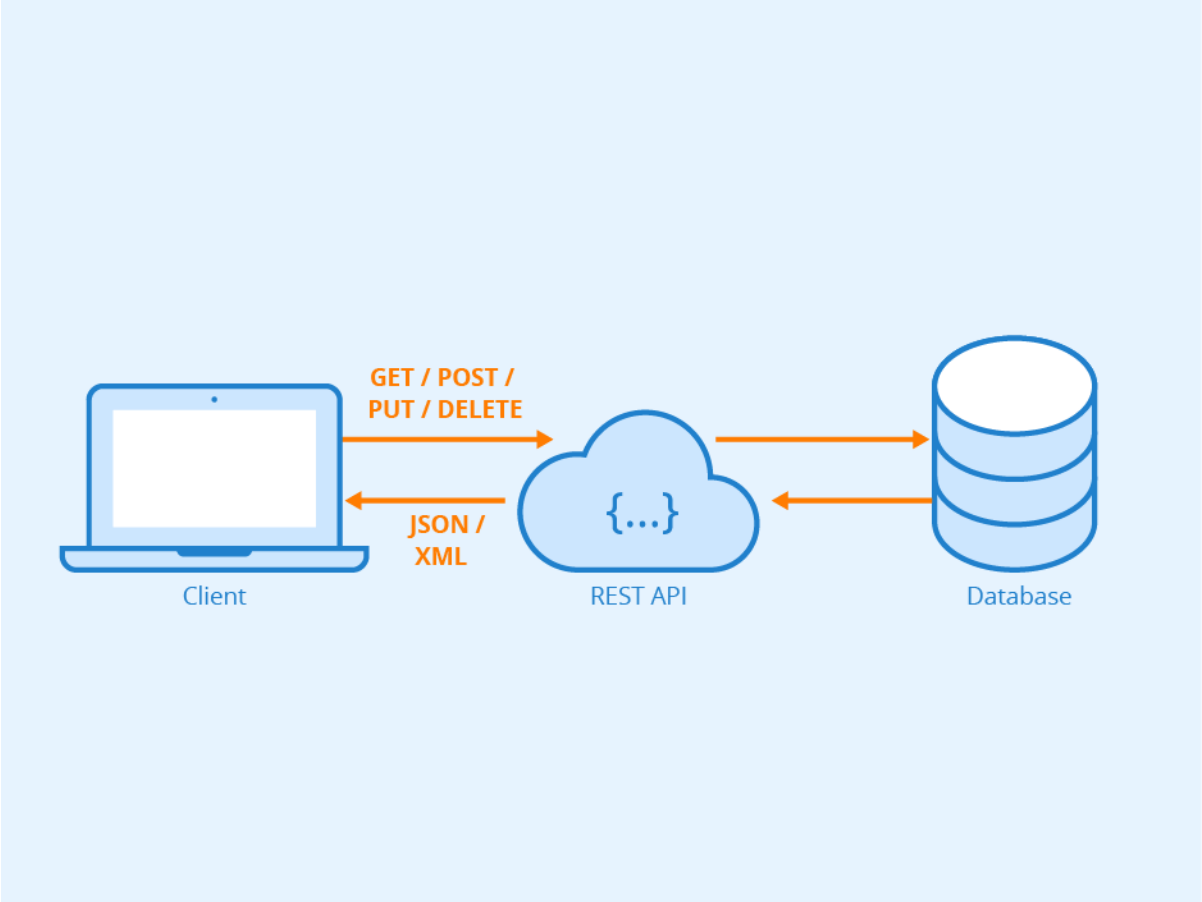


Рис. 6. Схема взаємодії через REST API

Авторизація та автентифікація

Розроблена система містить можливості авторизації та автентифікації що належать до виду Session Based Authentication, принцип дії якої зображено на рис. 7.

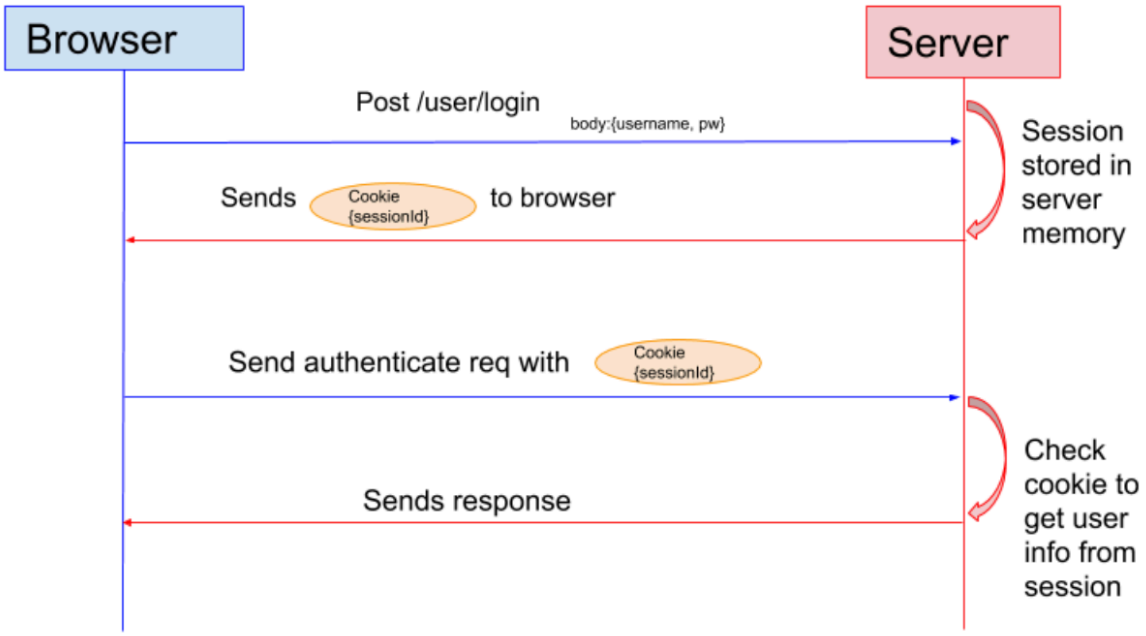


Рис. 7. Session Based Authentication

Механізм авторизації та автентифікації реалізовано за допомогою модуля passport.js.

Виділяють три типи користувачів:

- Visitor – будь-який користувач сайту, що не авторизувався у системі. Такі користувачі можуть лише переглядати контент веб-додатку, але не змінювати його;
- User – авторизований користувач для якого доступні функціональні можливості модифікації контенту веб-додатку;
- Admin – адміністратор, що має всі можливості звичайного користувача, а також може додавати та видаляти інших користувачів, які не є адміністраторами.

Реєстрація нових користувачів у системі проводиться тільки через адміністратора.

Стрічка новин RSS

Стрічка або канал RSS новин надає користувачам опис нової інформації, що з'явилася на сайті, а також посилання на її повну версію. Більшість сучасних веб-браузерів вміють працювати з RSS стрічками. Крім того існують спеціальні застосунки (RSS-агрегатори), які збирають та оброблюють дані з RSS каналів.

Для реалізації стрічки новин RSS був використаний npm модуль feed.

Приклад елементу стрічки новин наведений на рис. 8.

```
<item>
    <title>Магістратура 2020. Вступ</title>
    <link>http://pzks.fpm.kpi.ua/news/bf86345f-a178-4a51-805f-34c25cab3527</link>
    <guid>bf86345f-a178-4a51-805f-34c25cab3527</guid>
    <description>Студенти, які навчаються на іншому факультеті та мають бажання
    вступити на факультет прикладної математики та випускники минулих років мають
    подавати документи через Приймальну комісію КПІ ім. Ігоря Сікорського.</description>
</item>
```

Рис. 8. Елемент стрічки новин RSS

3.4. Висновки до розділу

У даному розділі був наведений загальний опис серверної частини розроблюваного веб-сайту кафедри програмного забезпечення комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Також було описано архітектуру системи та розкрито особливості її реалізації. У даному розділі наведено схему бази даних, детальний опис назв та типів полів таблиць та взаємозв'язків між ними. Було описано принцип взаємодії клієнтської та серверної частини за допомогою REST API, та особливості реалізації даної взаємодії при розробленні даної системи. Було описано реалізацію Session Based Authentication для забезпечення авторизації та автентифікації користувачів

системи за допомогою модуля passport.js. Також було описано реалізацію стрічки новин RSS, за допомогою модуля feed.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ

4.1. Аналіз реалізованого веб-додатку

Результатом розробки є веб-додаток, виконаний з урахуванням сучасних тенденцій розробки ПЗ. Основною функцією веб додатку є представлення інформації про кафедру у структурованому вигляді.

Відповідно до поставлених задач і вимог з боку серверної частини були представлені наступні рішення:

- використаний захищений протокол передачі даних;
- реалізовано RSS;
- впроваджена система редагування контенту;
- були перенесені дані зі старої версії веб-додатку;
- забезпечено взаємодію з базою даних;
- реалізовано взаємодію з клієнтською частиною;
- досягнуто високої швидкості роботи додатку.

Впроваджено систему реєстрації, авторизації та автентифікації користувачів. Авторизовані користувачі можуть:

- додавати інформацію про публікації до архіву публікації;
- додавати інформацію про випускні роботи;
- змінювати інформацію про співробітників кафедри;
- змінювати розклад консультацій;
- змінювати розклад екзаменів;
- змінювати інформацію про навчальний план;
- додавати новини у стрічку новин.

Адміністратори також можуть змінювати інформацію про інших користувачів системи. Взаємодія клієнтської і серверної частин реалізована за допомогою REST API. На рис. 9. наведено приклад типового RESTful API для модулю новин.

```
app.get('/api/news', db.getNews);
app.get('/api/news/:id', db.getNewsById);
app.post('/api/news', db.createNews);
app.put('/api/news/:id', db.updateNews);
app.delete('/api/news/:id', db.deleteNews);
```

Рис. 9. RESTful API для модулю новин

Програмний інтерфейс для інших модулів виконаний за аналогічним сценарієм. Взаємодія з базою даних відбувається за допомогою SQL-запитів.

За призначенням запити бувають наступних видів:

- SELECT – використовуються для читання даних з БД;
- INSERT – використовуються для додавання нових даних до БД;
- UPDATE – використовуються для оновлення наявних в БД даних;
- DELETE – використовуються для видалення даних з БД.

4.2. Тестування веб-додатку

Тестування є невід'ємною частиною процесу розробки ПЗ і виконується на кожному його етапі що сприяє виявленню помилок та недоліків на ранніх етапах, коли ціна їх усунення не надто висока. Тестування з залученням майбутніх користувачів є важливим у випадку коли вимоги були зібрані неправильно або є неповними, враховуючи відгуки користувачів допомагає точніше визначити потреби ринку та пріоритезувати розробку елементів системи. Головними цілями тестування є пошук помилок, перевірка зручності графічного інтерфейсу та перевірка коректності роботи програми в цілому.

У ході розробки веб-додатку тестування проводилось після реалізації кожного модуля системи що дозволило усунути помилки одразу після виявлення що унеможливило їх вплив на подальший процес розробки.

Після ручного тестування інтерфейсу користувачами були внесені деякі зміни до дизайну веб-додатку, що допомогло покращити сприйняття інформації та вигляд інтерфейсу в цілому.

Для перевірки правильності запитів, що надсилаються до сервера, а також перевірки відповідей від сервера використовувався браузер Google Chrome та вбудовані в нього інструменти для розробників.

Перевірка коректності записування, оновлення та видалення даних у базі даних проводилася за рахунок консольного інтерфейсу користувача, що входить в пакет поставки PostgreSQL.

Для тестування функціональних можливостей серверної частини були написані автоматичні тести, що в майбутньому дозволить легше виконувати рефакторинг та вносити зміни не завдаючи шкоди роботі інших компонентів системи.

У результаті тестування була отримана цінна інформація, яка допомогла знайти та виправити проблеми та недоліки у роботі веб-додатку, а також інформація що допомогла визначити шляхи майбутнього вдосконалення системи.

4.3. Порівняння розробки із існуючими аналогами

У підрозділі 1.2 дипломного проєкту було розглянуто веб-додатки вищих учбових закладів України та Європи. Був проведений аналіз переваг та недоліків аналогічних застосунків. Кожен з них розроблений відповідно до нормативних документів установ які вони представляють. Функціональні можливості аналогічних застосунків також різняться відповідно до потреб учбових закладів.

Головними критеріями оцінки були:

- сучасність дизайну – після опитування користувачів було визначено, що дизайн веб-додатку не виглядає застарілим;

- швидкодія – після тестування було визначено що швидкодія розробленого ПЗ нічим не поступається, а інколи і перевершує аналоги;
- зручність інтерфейсу – інтерфейс веб-додатку спроєктовано таким чином, щоб користувачі могли інтуїтивно зрозуміти де знайти потрібну інформацію;
- наявність захищеного протоколу передачі даних – розроблене програмний продукт підтримує HTTPS, що захищає користувачів від втрати їх особистих даних;
- крос-платформність – розроблене програмне забезпечення виконано у вигляді веб-додатку, що дозволяє використовувати його з будь-якого пристрою на якому встановлено веб-браузер з підтримкою JavaScript;
- наявність RSS – розроблене програмне забезпечення містить стрічку новин RSS, яка є загальноприйнятим форматом розповсюдження новин для різних веб-додатків, у тому числі й веб-додатків учбових закладів.

Отже, порівнявши аналоги у їх поточній реалізації можна стверджувати, що розроблений веб-сайт задовольняє поставлені вимоги і є гідним конкурентом.

4.4. Рекомендації щодо подальшого вдосконалення

Було визначено наступні шляхи для подальшого вдосконалення:

- реалізація форми зворотного зв’язку, що сприятиме покращенню комунікації між користувачами та адміністраторами веб-додатку;
- реалізація архіву матеріалів, що дозволить тримати всю матеріально-технічну базу в одному місці і допоможе вирішити проблему пошуку необхідних матеріалів студентами;

- покращення систем редагування контенту для адміністраторів веб-додатку за рахунок впровадження більш зручного інтерфейсу, що дозволить пришвидшити даний процес та уникнути помилок при модифікації;
- впровадження особистих кабінетів користувача, що допоможе організувати учбовий процес та комунікацію між співробітниками кафедри та студентами через веб-сайт. Наприклад, на сайті буде розміщено завдання які студенти мають виконати. Виконані завдання завантажуються на сайт студентами, викладачі їх перевіряють та виставляють оцінки, або відправляють завдання на доопрацювання;
- розширення можливостей інтерфейсу для людей з вадами зору: зміна розміру шрифту, контрастності та кольорової гами елементів інтерфейсу, тощо;
- реалізація модулю статистики, що допоможе слідкувати за відвідуваністю веб-додатку;
- додати систему захисту від DoS/DDoS-атак;
- забезпечення автоматичного розширення для витримування більших навантажень. В поточній реалізації граничне навантаження на систему залежить від потужності машини на якій виконується серверне ПЗ. При необхідності можна модернізувати систему, щоб при наближенні поточного навантаження на машину до граничного, запускалася копія сервісу на іншій машині і частина запитів користувачів перенаправлялася на неї;
- забезпечення шардингу та реплікаціїї бази даних, для збільшення її відмовостійкості та полегшення відновлення у разі відмови, а також збільшення навантаження яке БД здатна витримувати;

- створення інформаційних сторінок кафедри у різних соціальних мережах, та додання до інтерфейсу посилань на них, а також додання посилань на сайти співробітників кафедри, які безпосередньо пов’язані з організацією навчального процесу.

4.5. Висновки до розділу

У даному розділі було проведено аналіз реалізованого програмного забезпечення у вигляді веб-сайту кафедри програмного забезпечення комп’ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Після порівняння кінцевого продукту з аналогами, було визначено що він якісно перевищує більшість вітчизняних аналогів, та майже не поступається зарубіжним аналогам. Було визначено шляхи вдосконалення веб-сайту, серед яких є опції різні за складністю та часом виконання, аж до перетворення веб-сайту на повноцінну навчальну платформу. Деякі шляхи вдосконалення наразі є недоцільними через те, що поточне навантаження на систему є невисоким, проте зі збільшенням навантаження актуальність цих покращень також збільшується. Також було описано процес тестування що проводилося на кожному етапі розробки та містить як ручне так і автоматичне тестування, з зазначенням інструментів, що використовувалися для тестування тих чи інших частин системи.

ВИСНОВКИ

Метою даного дипломного проєкту було створення веб-сайту кафедри програмного забезпечення комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Перед розробкою був проведений аналіз нормативних документів, що регламентують роботу веб-додатків підрозділів університету, а також порівняльний аналіз веб-додатків інших учбових закладів. На основі проведеного аналізу були визначені вимоги до розроблюваного програмного забезпечення.

У даному проєкті проаналізовано та вибрано програмні засоби реалізації. Для реалізації серверної частину обрано мову програмування JavaScript та веб-фреймворк Express.js, реляційну базу даних PostgreSQL.

У дипломному проєкті розроблено архітектуру серверної частини веб-додатку, структуру бази даних, стрічку новин RSS. За рахунок впровадження системи авторизації та автентифікації реалізовано розділення функціональних можливостей залежно від типу користувачів, а також захист від несанкціонованого доступу.

Розробка програмного забезпечення виконана у відповідності до поставлених вимог у повному обсязі. Тестування системи проведено згідно зі затвердженою програмою та методикою тестування.

Пояснювальна записка також містить опис реалізованої системи та рекомендації щодо її вдосконалення.

Впровадження результатів розробки допоможе:

- покращити обмін інформацією між різними підрозділами університету;

- вирішити науково-інноваційні та освітні задач кафедри і університету за допомогою використання сучасних інформаційних технологій;
- оперативно та об'єктивно інформувати викладачів, студентів, аспірантів, співробітників, докторантів, випускників, абітурієнтів, ділових партнерів, а також інших зацікавлених осіб про різні аспекти діяльності кафедри;
- створити позитивний імідж кафедри, як університетського підрозділу, що здатен конкурувати на міжнародному ринку послуг у сфері науки та освіти.

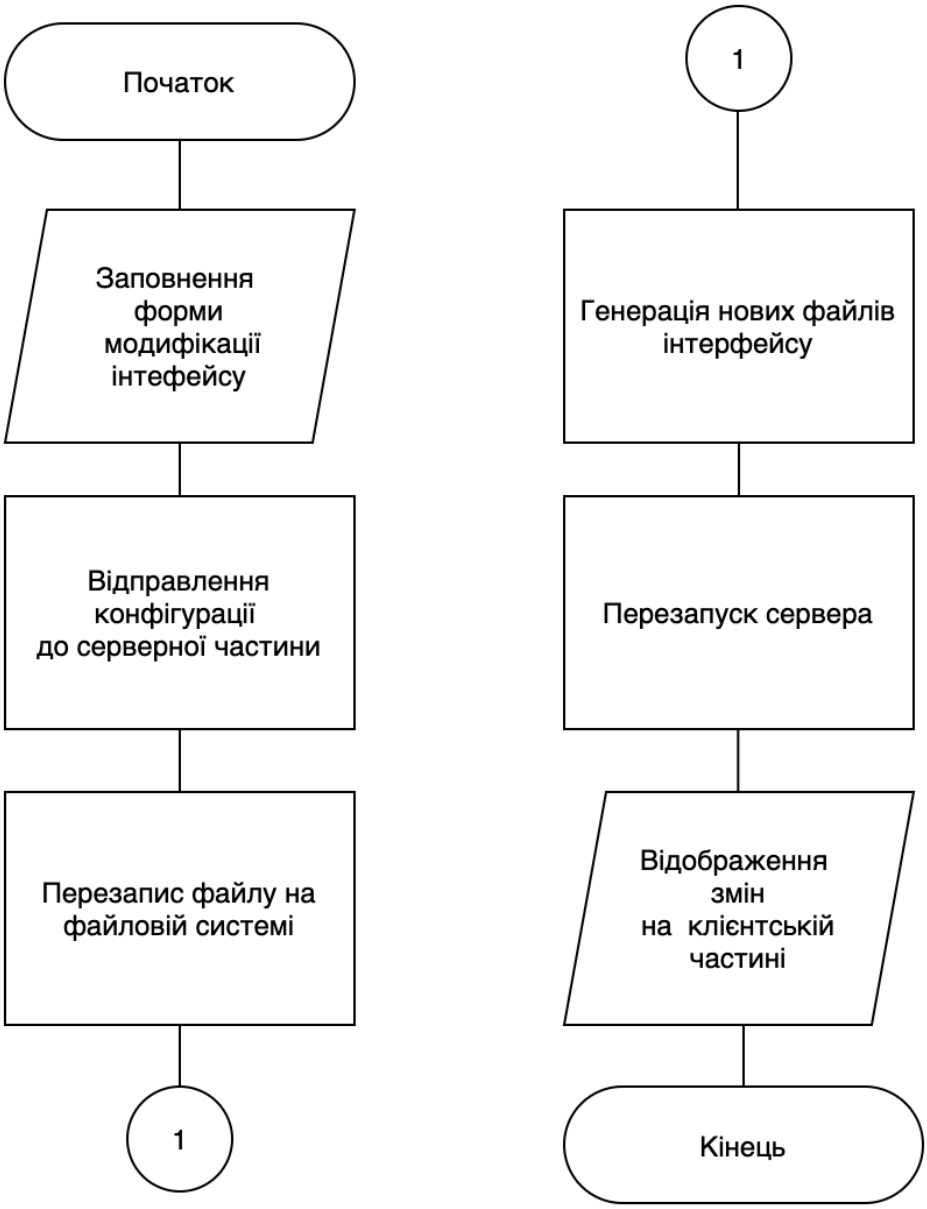
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Jagiellonian University – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://en.uj.edu.pl.
2. Technical University of Munich – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://www.tum.de.
3. НМУ ім. О.О. Богомольця – Головна [Електронний ресурс]. — Режим доступу до ресурсу: https://nmuofficial.com.
4. Python – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python.
5. C# – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/C_Sharp_(programming_language).
6. Java – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Java_(programming_language).
7. JavaScript – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript.
8. Advantages And Disadvantages of RDBMS [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/0lo37y.
9. An Overview of SQL and NoSQL with it’s Pros and Cons [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/4mo37y.
10. MySQL [Електронний ресурс]. — Режим доступу до ресурсу: https://searchoracle.techtarget.com/definition/MySQL.
11. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/ptp37y.
12. Express.js – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Express.js.

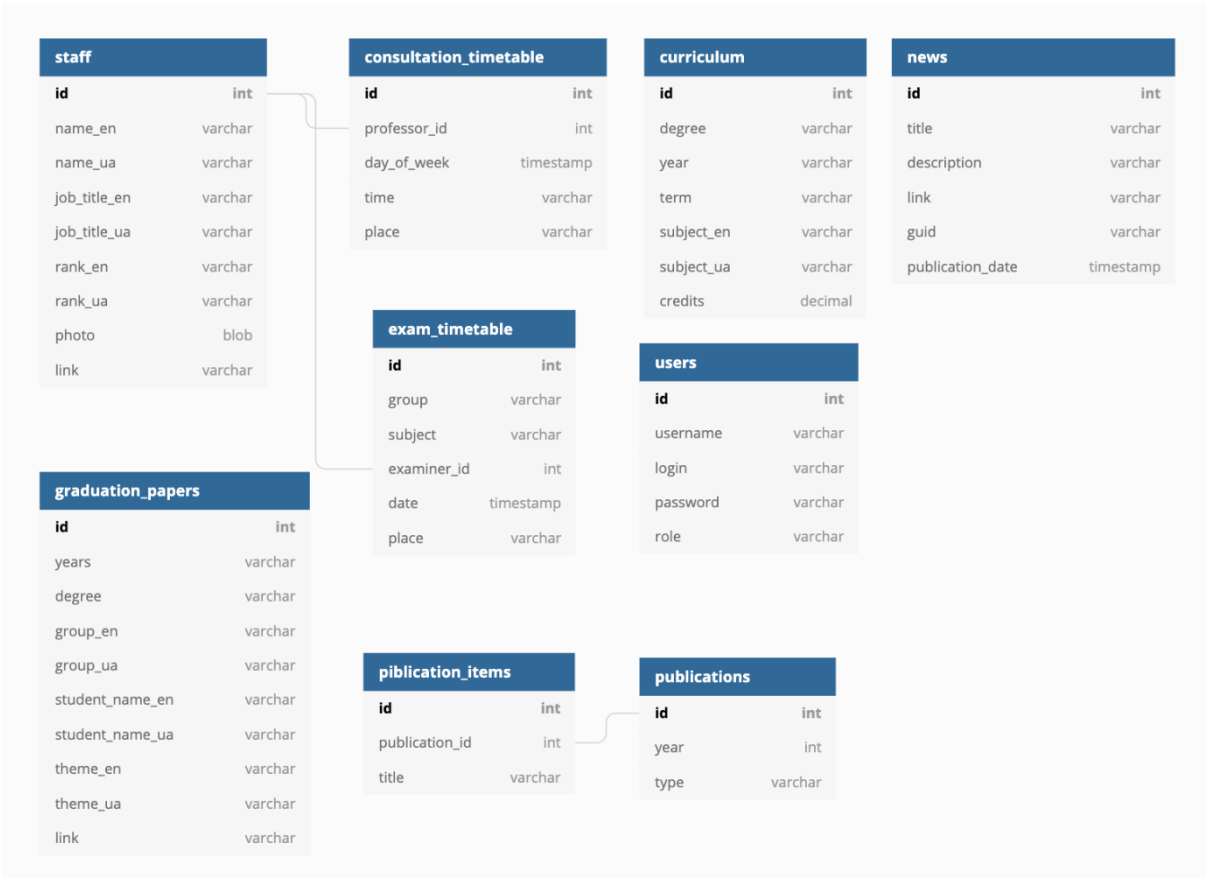
13. Meteor.js – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Meteor_(web_framework).
14. Client Server Architecture [Електронний ресурс]. — Режим доступу до ресурсу: https://cio-wiki.org/wiki/Client_Server_Architecture.
15. Chris Northwood. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer, 2018.
16. Hans-Jurgen Schonig. Mastering PostgreSQL 12: Advanced Techniques to Build and Administer Scalable and Reliable PostgreSQL Database Applications, 3rd Edition, 2019.
17. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack, 2014.
18. Ben Hammersley. Developing feeds with RSS and Atom, 2005.

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

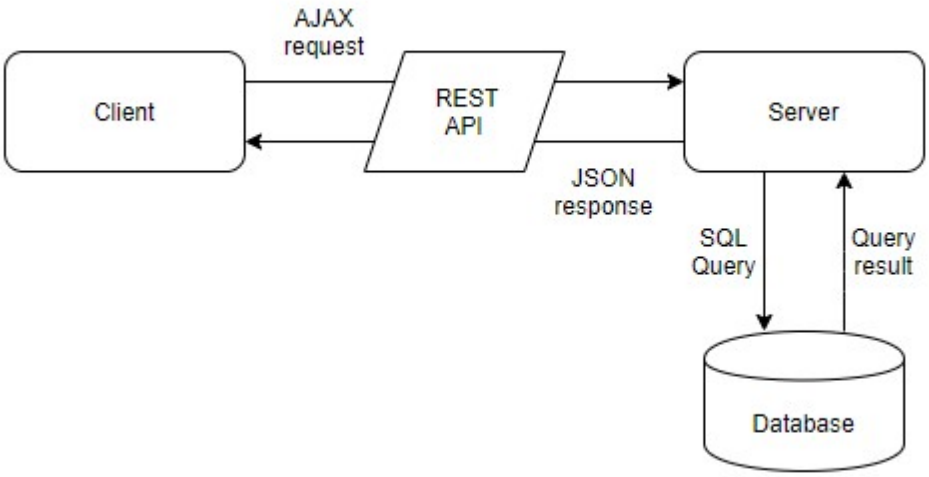


ДП.045440-06-99.
Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина. Алгоритм редагування інтерфейсу. Схема роботи алгоритму



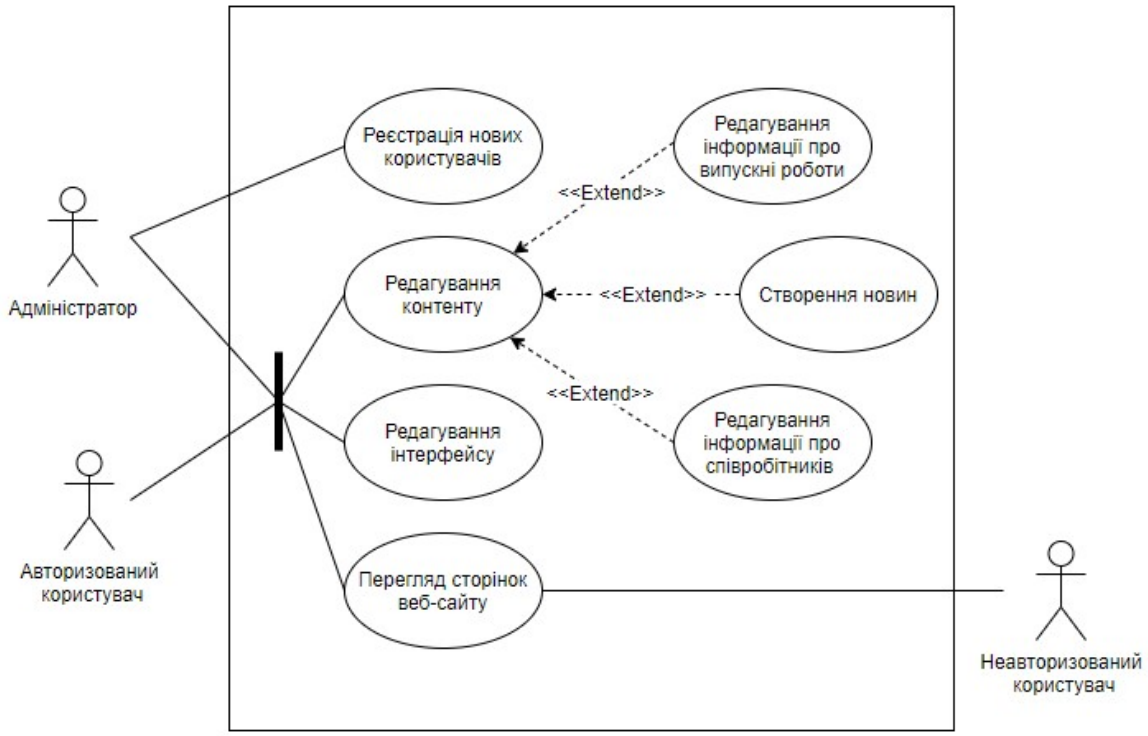
ДП.045440-07-99.
Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина. Структура бази даних. ERD-діаграма

Структурна схема системи



Півненко Олексій, група КП-61

Діаграма прецедентів



Півненко Олексій, група КП-61

Додаток 2
Лістинг програми

```
{
    "name": "pzks-website",
    "version": "1.0.0",
    "description": "FAM PZKS official website",
    "scripts": {
        "start:prod": "pm2 start npm --name \"server\" -- run
start:server",
        "start:dev": "webpack-dev-server --hot --open --mode development",
        "start:server": "node server.js",
        "build": "rm -rf dist && webpack --config
webpack.config.production.js ",
        "now-build": "npm run build",
        "test": "xo && stylelint './src/**/*.js'"
    },
    "main": "src/index.js",
    "repository": "https://github.com/xxczaki/styled-react-boilerplate",
    "author": "Egor Shchupakovskiy",
    "license": "MIT",
    "devDependencies": {
        "@babel/core": "^7.8.3",
        "@babel/preset-env": "^7.8.3",
        "@babel/preset-react": "^7.8.3",
        "@pieh/friendly-errors-webpack-plugin": "^1.7.0-chalk-2",
        "babel-eslint": "^10.0.3",
        "babel-loader": "^8.0.6",
        "babel-plugin-styled-components": "^1.10.6",
        "clean-css-loader": "^2.0.0",
        "css-loader": "^3.4.2",
        "eslint-config-xo-react": "^0.22.0",
        "eslint-plugin-react": "^7.18.0",
        "eslint-plugin-react-hooks": "^2.3.0",
        "extract-css-chunks-webpack-plugin": "^4.7.4",
        "file-loader": "^5.0.2",
        "html-webpack-plugin": "^3.2.0",
        "imagemin": "^7.0.1",
        "imagemin-mozjpeg": "^8.0.0",
        "imagemin-pngquant": "^8.0.0",
        "imagemin-svgo": "^7.0.0",
        "imagemin-webp": "^5.1.0",
        "img-loader": "^3.0.1",
        "json-loader": "^0.5.7",
        "script-ext-html-webpack-plugin": "^2.1.4",
        "style-loader": "^1.2.1",
        "stylelint": "^13.0.0",
        "stylelint-config-recommended": "^3.0.0",
        "stylelint-config-styled-components": "^0.1.1",
        "stylelint-processor-styled-components": "^1.9.0",
        "terser-webpack-plugin": "^2.3.2",
        "url-loader": "^3.0.0",
        "webpack": "^4.41.5",
        "webpack-cli": "^3.3.10",
        "webpack-dev-server": "^3.10.1",
        "webpack-nomodule-plugin": "^1.0.1",
        "webpack-pwa-manifest": "^4.1.1",
        "workbox-webpack-plugin": "^4.3.1",
        "xo": "^0.25.3"
    },
    "dependencies": {
        "@fortawesome/fontawesome-svg-core": "^1.2.28",
        "@fortawesome/free-solid-svg-icons": "^5.13.0",
        "@fortawesome/react-fontawesome": "^0.1.10",
        "@hot-loader/react-dom": "^16.11.0",
        "axios": "^0.19.2",
        "body-parser": "^1.19.0",
        "compression": "^1.7.4",
```

```
        "core-js": "^3.6.4",
        "express": "^4.17.1",
        "express-basic-auth": "^1.2.0",
        "express-history-api-fallback": "^2.2.1",
        "feed": "^4.2.0",
        "formik": "^2.1.4",
        "fs": "0.0.1-security",
        "history": "^4.10.1",
        "i18n": "^0.8.6",
        "i18next": "^19.4.1",
        "i18next-browser-languagedetector": "^4.0.2",
        "interweave": "^12.5.0",
        "lodash": "^4.17.15",
        "modern-normalize": "^0.6.0",
        "moment": "^2.26.0",
        "normalize.css": "^8.0.1",
        "pg": "^8.2.1",
        "react": "^16.13.1",
        "react-anchor-link-smooth-scroll": "^1.0.12",
        "react-dom": "^16.12.0",
        "react-ga": "^3.0.0",
        "react-hot-loader": "^4.12.19",
        "react-i18next": "^11.3.4",
        "react-json-view": "^1.19.1",
        "react-router-dom": "^5.1.2",
        "semantic-ui-css": "^2.4.1",
        "semantic-ui-react": "^0.88.2",
        "shelljs": "^0.8.4",
        "styled-components": "^5.0.0"
    },
    "xo": {
        "nodeVersion": ">=10",
        "parser": "babel-eslint",
        "envs": [
            "node",
            "browser"
        ],
        "extends": "xo-react",
        "settings": {
            "react": {
                "version": "16.12"
            }
        },
        "rules": {
            "import/no-unassigned-import": "off"
        }
    }
}
const Feed = require('feed').Feed;
var fs = require('fs');
var dirPath = __dirname + "/public/rss.xml";

const feedOptions = {
    title: "Кафедра програмного забезпечення комп'ютерних систем",
    link: "http://pzks.fpm.kpi.ua/",
    description : "",
    language: "UK",
};

var feed = new Feed(feedOptions);

const createNewFeed = () => {
    feed = new Feed(feedOptions);
}
```

```
const loadRss = () => {
    return fs.readFileSync(dirPath, 'utf8')
}

const addFeedItem = (item) => {
    feed.addItem(item);
}

const saveRss = () => {
  let rssdoc = feed.rss2();
  fs.writeFile(dirPath, rssdoc, function (err) {
    if (err) {
      throw err;
    }
  });
}


module.exports = {
    loadRss,
    saveRss,
    addFeedItem,
    createNewFeed
  }
const express = require("express");
const fs = require("fs");
const fallback = require("express-history-api-fallback");
const compression = require("compression");
const { exec } = require("child_process");
const basicAuth = require("express-basic-auth");
var bodyParser = require("body-parser");

const db = require("./queries");
const rss = require("./rss");

const app = express();
const port = process.env.PORT || 8080;

app.use(
  bodyParser.json({
    limit: "50mb",
  })
);

app.use(
  bodyParser.urlencoded({
    limit: "50mb",
    parameterLimit: 100000,
    extended: true,
  })
);

app.use(
  "/admin",
  basicAuth({
    challenge: true,
    authorizer,
    unauthorizedResponse: (req) => {
      return `unauthorized. ip: ${req.ip}`;
    },
  })
);
function authorizer(username, password) {
  const userMatches = basicAuth.safeCompare(username, "admin");
  const passwordMatches = basicAuth.safeCompare(password, "admin");
```

```
  return userMatches & passwordMatches;
}

app.get("/api/news", db.getNews);
app.get("/api/news/:id", db.getNewsById);
app.post("/api/news", db.createNews);
app.put("/api/news/:id", db.updateNews);
app.delete("/api/news/:id", db.deleteNews);

app.get("/rss", (req, res) => {
  rssFeed = rss.loadRss();
  res.type("rss");
  res.send(rssFeed);
});

app.put("/api/admin", (req, res) => {
  const { content, type } = req.body;
  const data = JSON.stringify(content, null, 2);
  switch (type) {
    case "UA":
      fs.writeFileSync(__dirname + "/src/i18n/locales/ua.json", data);
      break;
    case "EN":
      fs.writeFileSync(__dirname + "/src/i18n/locales/en.json", data);
      break;
    case "curriculum":
      fs.writeFileSync(
        __dirname + "/src/Project/CurriculumInfo/curriculumInfoData.json",
        data
      );
      break;
    case "exam":
      fs.writeFileSync(
        __dirname +
"/src/Project/EducationalProcess/Schedules/exam/exams.json",
        data
      );
      break;
    case "consultation":
      fs.writeFileSync(
        __dirname +

"/src/Project/EducationalProcess/Schedules/consultation/consultation.json",
        data
      );
      break;
    case "graduationPaper":
      fs.writeFileSync(
        __dirname +

"/src/Project/EducationalProcess/GraduationPaper/graduationPapers.json",
        data
      );
      break;
    default:
      return;
  }
  exec("./restart_server.sh");
});

app.get("/api/isAdmin", (req, res) => {
  if (req.headers.authorization) {
    res.json({ isAdmin: true });
  } else {
```

```
    res.json({ isAdmin: false });
  }
});

app.use(compression());

app.use(express.static(`${__dirname}/dist`));

app.use(fallback(`${__dirname}/dist/index.html`));

app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
      var t = e.active,
        n = e.children,
        a = e.className,
        o = e.collapsing,
        i = e.content,
        l = e.disabled,
        u = e.error,
        g = e.icon,
        h = e.negative,
        A = e.positive,
        M = e.selectable,
        v = e.singleLine,
        I = e.textAlign,
        N = e.verticalAlign,
        C = e.warning,
        j = e.width,
        k = s()(
          Object(d.a)(t, "active"),
          Object(d.a)(o, "collapsing"),
          Object(d.a)(l, "disabled"),
          Object(d.a)(u, "error"),
          Object(d.a)(h, "negative"),
          Object(d.a)(A, "positive"),
          Object(d.a)(M, "selectable"),
          Object(d.a)(v, "single line"),
          Object(d.a)(C, "warning"),
          Object(d.d)(I),
          Object(d.f)(N),
          Object(d.g)(j, "wide"),
          a
        ),
        _ = Object(p.a)(b, e),
        L = Object(f.a)(b, e);
      return m.a.isNil(n)
        ? c.a.createElement(L, r()({}, _, { className: k }), y.a.create(g),
i)
        : c.a.createElement(L, r()({}, _, { className: k }), n);
    }
    (b.handledProps = [
      "active",
      "as",
      "children",
      "className",
      "collapsing",
      "content",
      "disabled",
      "error",
      "icon",
      "negative",
      "positive",
      "selectable",
      "singleLine",
```

```
      "textAlign",
      "verticalAlign",
      "warning",
      "width",
    ]),
      (b.defaultProps = { as: "td" }),
      (b.propTypes = {}),
      (b.create = Object(A.e)(b, function(e) {
        return { content: e };
      }));
    var M = b;
    function v(e) {
      var t = e.children,
        n = e.className,
        a = e.content,
        o = e.fullWidth,
        i = s()(Object(d.a)(o, "full-width"), n),
        l = Object(p.a)(v, e),
        u = Object(f.a)(v, e);
      return c.a.createElement(
        u,
        r()({}, l, { className: i }),
        m.a.isNil(t) ? a : t
      );
    }
    (v.handledProps = ["as", "children", "className", "content",
"fullWidth"]),
      (v.defaultProps = { as: "thead" }),
      (v.propTypes = {});
    var I = v;
    function N(e) {
      var t = e.as,
        n = Object(p.a)(N, e);
      return c.a.createElement(I, r()({}, n, { as: t }));
    }
    (N.handledProps = ["as"]),
      (N.propTypes = {}),
      (N.defaultProps = { as: "tfoot" });
    var C = N;
    function j(e) {
      var t = e.as,
        n = e.className,
        a = e.sorted,
        o = s()(Object(d.e)(a, "sorted"), n),
        i = Object(p.a)(j, e);
      return c.a.createElement(M, r()({}, i, { as: t, className: o }));
    }
    (j.handledProps = ["as", "className", "sorted"]),
      (j.propTypes = {}),
      (j.defaultProps = { as: "th" });
    var k = j;
    function _(e) {
      var t = e.active,
        n = e.cellAs,
        a = e.cells,
        o = e.children,
        l = e.className,
        u = e.disabled,
        g = e.error,
        h = e.negative,
        A = e.positive,
        y = e.textAlign,
        b = e.verticalAlign,
        v = e.warning,
        I = s()(
```

```
      Object(d.a)(t, "active"),
      Object(d.a)(u, "disabled"),
      Object(d.a)(g, "error"),
      Object(d.a)(h, "negative"),
      Object(d.a)(A, "positive"),
      Object(d.a)(v, "warning"),
      Object(d.d)(y),
      Object(d.f)(b),
      l
    ),
    N = Object(p.a)(_, e),
    C = Object(f.a)(_, e);
  return m.a.isNil(o)
    ? c.a.createElement(
        C,
        r()({}, N, { className: I }),
        i()(a, function(e) {
          return M.create(e, { defaultProps: { as: n } });
        })
      )
    : c.a.createElement(C, r()({}, N, { className: I }), o);
}
(_.handledProps = [
  "active",
  "as",
  "cellAs",
  "cells",
  "children",
  "className",
  "disabled",
  "error",
  "negative",
  "positive",
  "textAlign",
  "verticalAlign",
  "warning",
]),
  (_.defaultProps = { as: "tr", cellAs: "td" }),
  (_.propTypes = {}),
  (_.create = Object(A.e)(_, function(e) {
    return { cells: e };
  }));
var L = _;
function T(e) {
  var t = e.attached,
    n = e.basic,
    a = e.celled,
    o = e.children,
    l = e.className,
    u = e.collapsing,
    g = e.color,
    A = e.columns,
    y = e.compact,
    b = e.definition,
    M = e.fixed,
    v = e.footerRow,
    N = e.headerRow,
    j = e.headerRows,
    k = e.inverted,
    _ = e.padded,
    E = e.renderBodyRow,
    D = e.selectable,
    U = e.singleLine,
    w = e.size,
    S = e.sortable,
```

```
    x = e.stackable,
    O = e.striped,
    Y = e.structured,
    z = e.tableData,
    F = e.textAlign,
    Q = e.unstackable,
    P = e.verticalAlign,
    V = s()(
      "ui",
      g,
      w,
      Object(d.a)(a, "celled"),
      Object(d.a)(u, "collapsing"),
      Object(d.a)(b, "definition"),
      Object(d.a)(M, "fixed"),
      Object(d.a)(k, "inverted"),
      Object(d.a)(D, "selectable"),
      Object(d.a)(U, "single line"),
      Object(d.a)(S, "sortable"),
      Object(d.a)(x, "stackable"),
      Object(d.a)(O, "striped"),
      Object(d.a)(Y, "structured"),
      Object(d.a)(Q, "unstackable"),
      Object(d.b)(t, "attached"),
      Object(d.b)(n, "basic"),
      Object(d.b)(y, "compact"),
      Object(d.b)(_, "padded"),
      Object(d.d)(F),
      Object(d.f)(P),
      Object(d.g)(A, "column"),
      "table",
      l
    ),
    H = Object(p.a)(T, e),
    R = Object(f.a)(T, e);
  if (!m.a.isNil(o))
    return c.a.createElement(R, r()({}, H, { className: V }), o);
  var W = { defaultProps: { cellAs: "th" } },
    B =
      (N || j) &&
      c.a.createElement(
        I,
        null,
        L.create(N, W),
        i()(j, function(e) {
          return L.create(e, W);
        })
      );
  return c.a.createElement(
    R,
    r()({}, H, { className: V }),
    B,
    c.a.createElement(
      h,
      null,
      E &&
        i()(z, function(e, t) {
          return L.create(E(e, t));
        })
    ),
    v && c.a.createElement(C, null, L.create(v))
  );
}
(T.handledProps = [
  "as",
```

```
    "attached",
    "basic",
    "celled",
    "children",
    "className",
    "collapsing",
    "color",
    "columns",
    "compact",
    "definition",
    "fixed",
    "footerRow",
    "headerRow",
    "headerRows",
    "inverted",
    "padded",
    "renderBodyRow",
    "selectable",
    "singleLine",
    "size",
    "sortable",
    "stackable",
    "striped",
    "structured",
    "tableData",
    "textAlign",
    "unstackable",
    "verticalAlign",
  ]),
    (T.defaultProps = { as: "table" }),
    (T.propTypes = {}),
    (T.Body = h),
    (T.Cell = M),
    (T.Footer = C),
    (T.Header = I),
    (T.HeaderCell = k),
    (T.Row = L);
  t.a = T;
},
function(e, t, n) {
  "use strict";
  var a = n(2),
    r = n.n(a),
    o = n(15),
    i = n.n(o),
    l = n(16),
    s = n.n(l),
    u = n(19),
    c = n.n(u),
    d = n(17),
    p = n.n(d),
    f = n(5),
    m = n.n(f),
    g = n(20),
    h = n.n(g),
    A = n(1),
    y = n.n(A),
    b = n(29),
    M = n.n(b),
    v = n(11),
    I = n.n(v),
    N = n(7),
    C = n.n(N),
    j = (n(10), n(0)),
    k = n.n(j),
```

```
  _ = n(42),
  L = n(152),
  T = n(153),
  E = n(6),
  D = n(238);
function U(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = C()(n, "description"),
    i = Object(L.a)(U, e),
    l = Object(T.a)(U, e);
  return k.a.createElement(
    l,
    r()({}, i, { className: o }),
    E.a.isNil(t) ? a : t
  );
}
(U.handledProps = ["as", "children", "className", "content"]),
  (U.propTypes = {}),
  (U.create = Object(D.e)(U, function(e) {
    return { content: e };
  }));
var w = U;
function S(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = C()("header", n),
    i = Object(L.a)(S, e),
    l = Object(T.a)(S, e);
  return k.a.createElement(
    l,
    r()({}, i, { className: o }),
    E.a.isNil(t) ? a : t
  );
}
(S.handledProps = ["as", "children", "className", "content"]),
  (S.propTypes = {}),
  (S.create = Object(D.e)(S, function(e) {
    return { content: e };
  }));
var x = S;
function O(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = e.description,
    i = e.floated,
    l = e.header,
    s = e.verticalAlign,
    u = C()(Object(_.e)(i, "floated"), Object(_.f)(s), "content", n),
    c = Object(L.a)(O, e),
    d = Object(T.a)(O, e);
  return E.a.isNil(t)
    ? k.a.createElement(
        d,
        r()({}, c, { className: u }),
        x.create(l),
        w.create(o),
        a
      )
    : k.a.createElement(d, r()({}, c, { className: u }), t);
}
(O.handledProps = [
```

```
      "as",
      "children",
      "className",
      "content",
      "description",
      "floated",
      "header",
      "verticalAlign",
    ]),
      (O.propTypes = {}),
      (O.create = Object(D.e)(O, function(e) {
        return { content: e };
      }));
    var Y = O,
      z = n(34);
    function F(e) {
      var t = e.className,
        n = e.verticalAlign,
        a = C()(Object(_.f)(n), t),
        o = Object(L.a)(F, e);
      return k.a.createElement(z.a, r()({}, o, { className: a }));
    }
    (F.handledProps = ["className", "verticalAlign"]),
      (F.propTypes = {}),
      (F.create = Object(D.e)(F, function(e) {
        return { name: e };
      }));
    var Q = F,
      P = n(150),
      V = n.n(P),
      H = n(156),
      R = (function(e) {
        function t() {
          var e, n;
          i()(this, t);
          for (var a = arguments.length, r = new Array(a), o = 0; o < a;
o++)
            r[o] = arguments[o];
          return (
            (n = c()(this, (e = p()(t)).call.apply(e, [this].concat(r)))),
            y()(m()(n), "handleClick", function(e) {
              n.props.disabled || I()(n.props, "onClick", e, n.props);
            }),
            n
          );
        }
        return (
          h()(t, e),
          s()(t, [
            {
              key: "render",
              value: function() {
                var e = this.props,
                  n = e.active,
                  a = e.children,
                  o = e.className,
                  i = e.content,
                  l = e.description,
                  s = e.disabled,
                  u = e.header,
                  c = e.icon,
                  d = e.image,
                  p = e.value,
                  f = Object(T.a)(t, this.props),
                  m = C()(
```

```
    Object(_.a)(n, "active"),
    Object(_.a)(s, "disabled"),
    Object(_.a)("li" !== f, "item"),
    o
  ),
  g = Object(L.a)(t, this.props),
  h = "li" === f ? { value: p } : { "data-value": p };
if (!E.a.isNil(a))
  return k.a.createElement(
    f,
    r()(
      {},
      h,
      {
        role: "listitem",
        className: m,
        onClick: this.handleClick,
      },
      g
    ),
    a
  );
var A = Q.create(c, { autoGenerateKey: !1 }),
  y = H.a.create(d, { autoGenerateKey: !1 });
if (!Object(j.isValidElement)(i) && V()(i))
  return k.a.createElement(
    f,
    r()(
      {},
      h,
      {
        role: "listitem",
        className: m,
        onClick: this.handleClick,
      },
      g
    ),
    A || y,
    Y.create(i, {
      autoGenerateKey: !1,
      defaultProps: { header: u, description: l },
    })
  );
var b = x.create(u, { autoGenerateKey: !1 }),
  M = w.create(l, { autoGenerateKey: !1 });
return A || y
  ? k.a.createElement(
      f,
      r()(
        {},
        h,
        {
          role: "listitem",
          className: m,
          onClick: this.handleClick,
        },
        g
      ),
      A || y,
      (i || b || M) && k.a.createElement(Y, null, b, M, i)
    )
  : k.a.createElement(
      f,
      r()(
        {},
```

```
                        h,
                        {
                          role: "listitem",
                          className: m,
                          onClick: this.handleClick,
                        },
                        g
                      ),
                      b,
                      M,
                      i
                    );
                },
              },
            ]),
            t
          );
        })(j.Component);
      y()(R, "handledProps", [
        "active",
        "as",
        "children",
        "className",
        "content",
        "description",
        "disabled",
        "header",
        "icon",
        "image",
        "onClick",
        "value",
      ]),
        (R.propTypes = {}),
        (R.create = Object(D.e)(R, function(e) {
          return { content: e };
        }));
      var W = R;
      function B(e) {
        var t = e.children,
          n = e.className,
          a = e.content,
          o = Object(L.a)(B, e),
          i = Object(T.a)(B, e),
          l = C()(Object(_.a)("ul" !== i && "ol" !== i, "list"), n);
        return k.a.createElement(
          i,
          r()({}, o, { className: l }),
          E.a.isNil(t) ? a : t
        );
      }
      (B.handledProps = ["as", "children", "className", "content"]),
        (B.propTypes = {});
      var Z = B,
        G = (function(e) {
          function t() {
            var e, n;
            i()(this, t);
            for (var a = arguments.length, r = new Array(a), o = 0; o < a;
o++)
              r[o] = arguments[o];
            return (
              (n = c()(this, (e = p()(t)).call.apply(e, [this].concat(r)))),
              y()(m()(n), "handleItemOverrides", function(e) {
                return {
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Виконав: Півненко Олексій Володимирович

Керівник: старший викладач кафедри ПЗКС, к.т.н., Люшенко Л.А.

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити серверну частину веб-сайту кафедри програмного забезпечення комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Завдання:

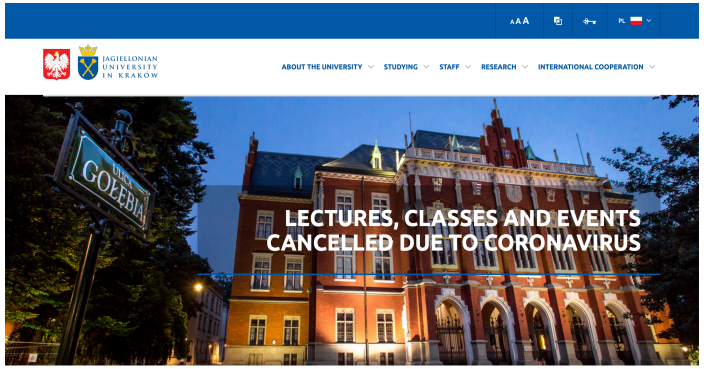
1. Проаналізувати нормативні документи, що регламентують роботу сайтів підрозділів університету, а також існуючі програмні аналоги та визначити вимоги до розроблюваного веб-сайту.
2. Розробити серверну частину веб-сайту.
3. Налаштувати взаємодію з клієнтською частиною.
4. Протестувати роботу системи.

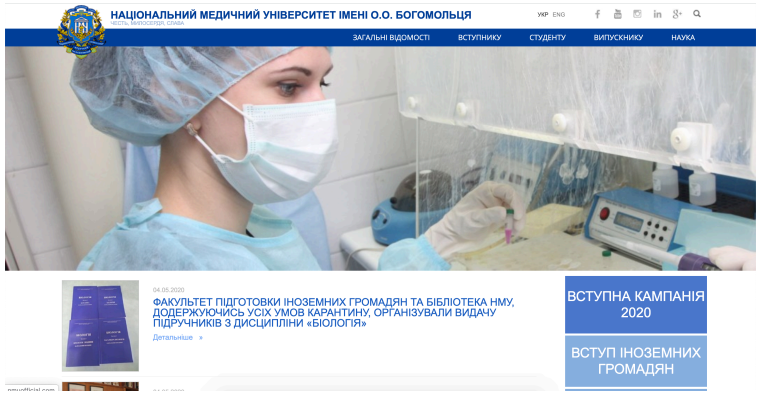
АКТУАЛЬНІСТЬ

Поточна версія веб-сайту містить наступні недоліки:

- структура веб-сайту не відповідає вимогам описаним у положенні про типовий сайт кафедри НТУУ «КПІ» у повному обсязі;
- застарілий дизайн;
- використання незахищеного протоколу передачі даних;
- відсутність стрічки новин RSS.

РОЗГЛЯНУТІ АНАЛОГИ





ВИМОГИ

- структура веб-сайту має бути розроблена згідно положення про типовий сайт кафедри НТУУ «КПІ»;
- забезпечення безперервного доступу користувачів;
- висока швидкість роботи;
- використання HTTPS;
- забезпечення можливості редагування інформації;
- коректна обробка будь-яких запитів користувача;
- наявність працюючого RSS.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

СХЕМА АРХІТЕКТУРИ СИСТЕМИ

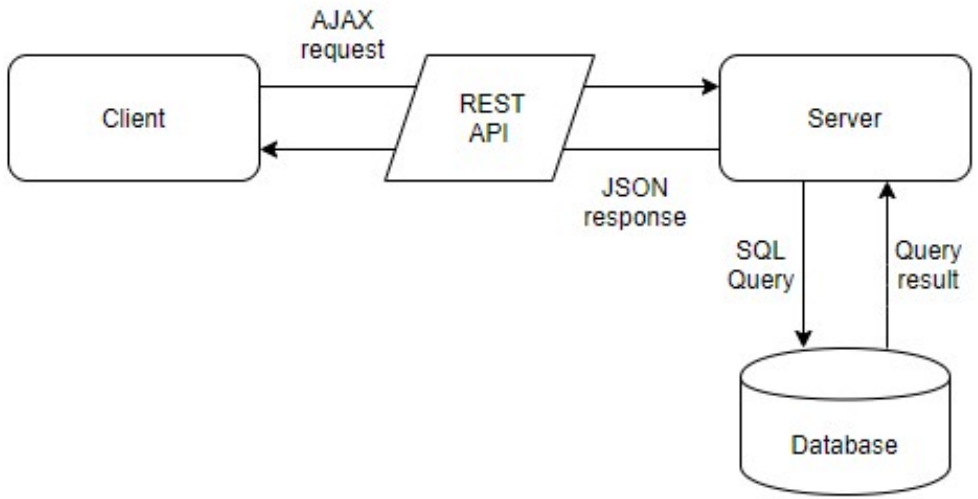
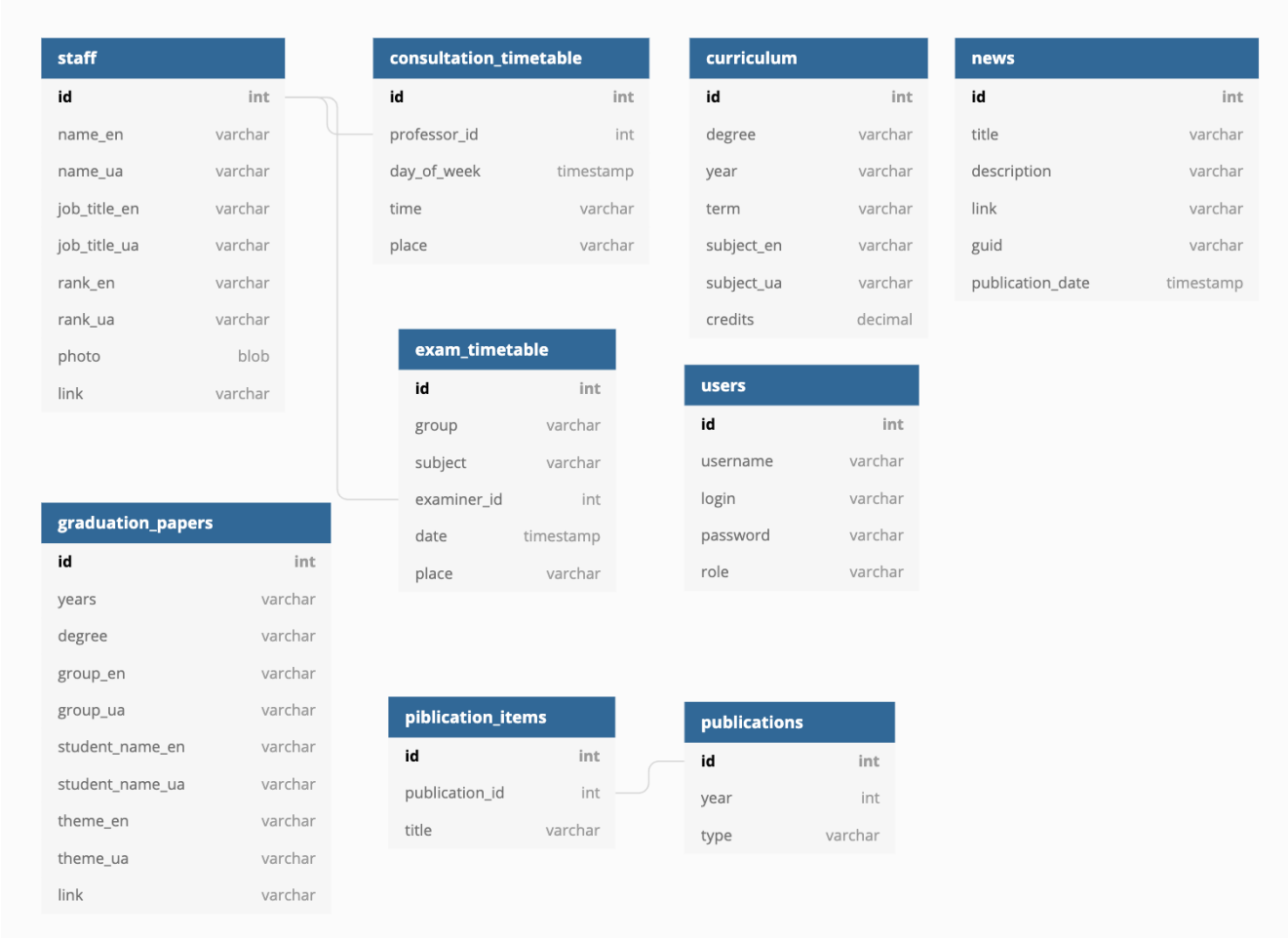
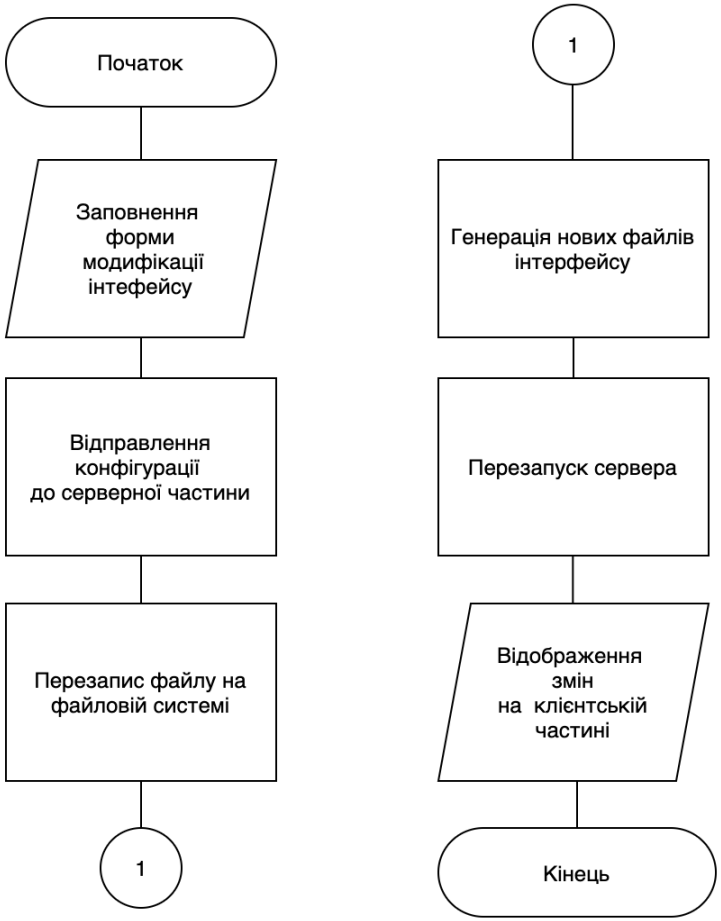


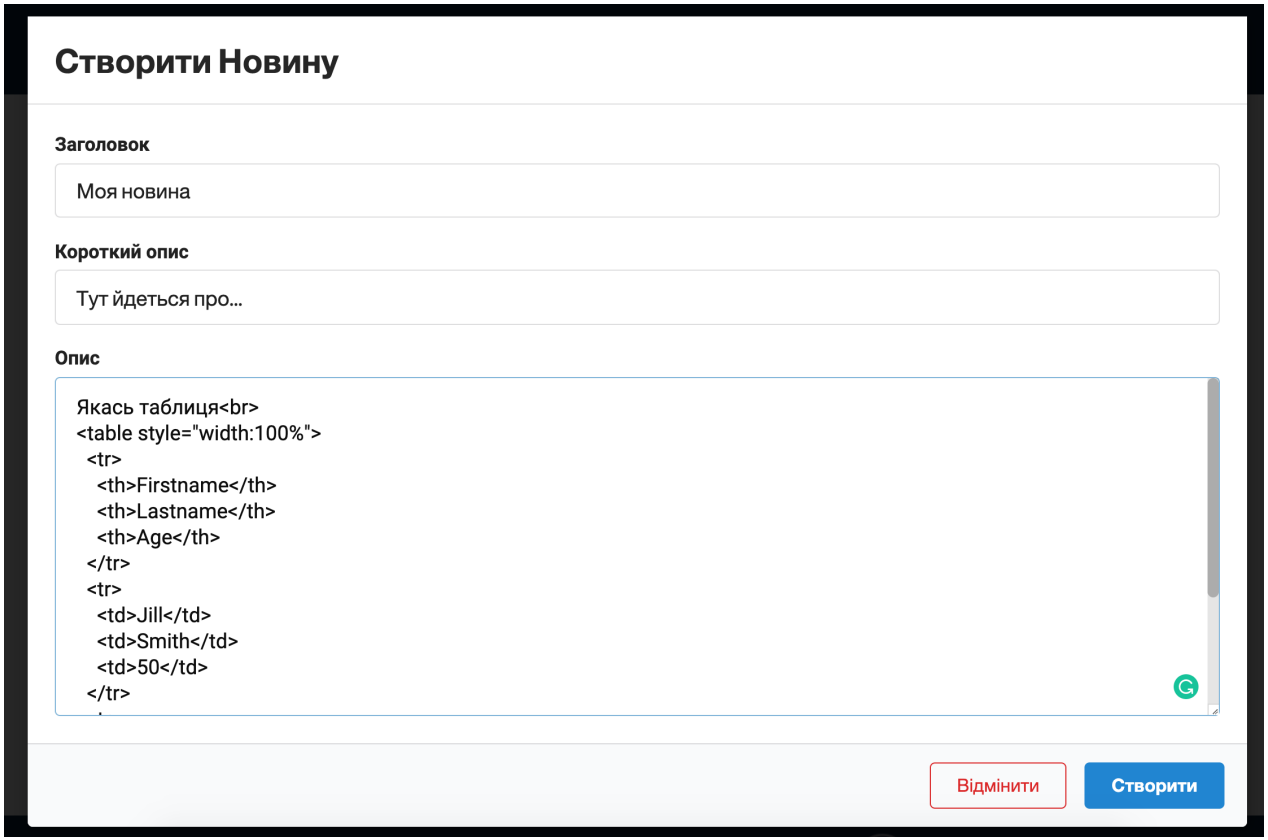
СХЕМА БАЗИ ДАНИХ



АЛГОРИТМ РЕДАГУВАННЯ ІНТЕРФЕЙСУ



СТВОРЕННЯ НОВИНИ



СТВОРЕННЯ НОВИНИ

Моя новина
17-06-2020
Тут йдеться про...

Детальніше

Моя новина

Якась таблиця

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

ПАНЕЛЬ АДМІНІСТРАТОРА

Розклад іспитів

"root" : [18 items
0 : {...} 6 items
1 : {...} 6 items
2 : {...} 6 items
3 : {...} 6 items
4 : {...} 6 items
5 : {...} 6 items
6 : {...} 6 items
7 : {...} 6 items
8 : {...} 6 items
9 : {...} 6 items
10 : {...} 6 items
11 : {...} 6 items
12 : {...} 6 items
13 : {...} 6 items
14 : {...} 6 items
15 : {...} 6 items
16 : {...} 6 items
17 : {...} 6 items
]

Застосувати зміни

Випускні кваліфікаційні роботи

"root" : [...] 6 items

Застосувати зміни

Навчальний план

"root" : [...] 3 items

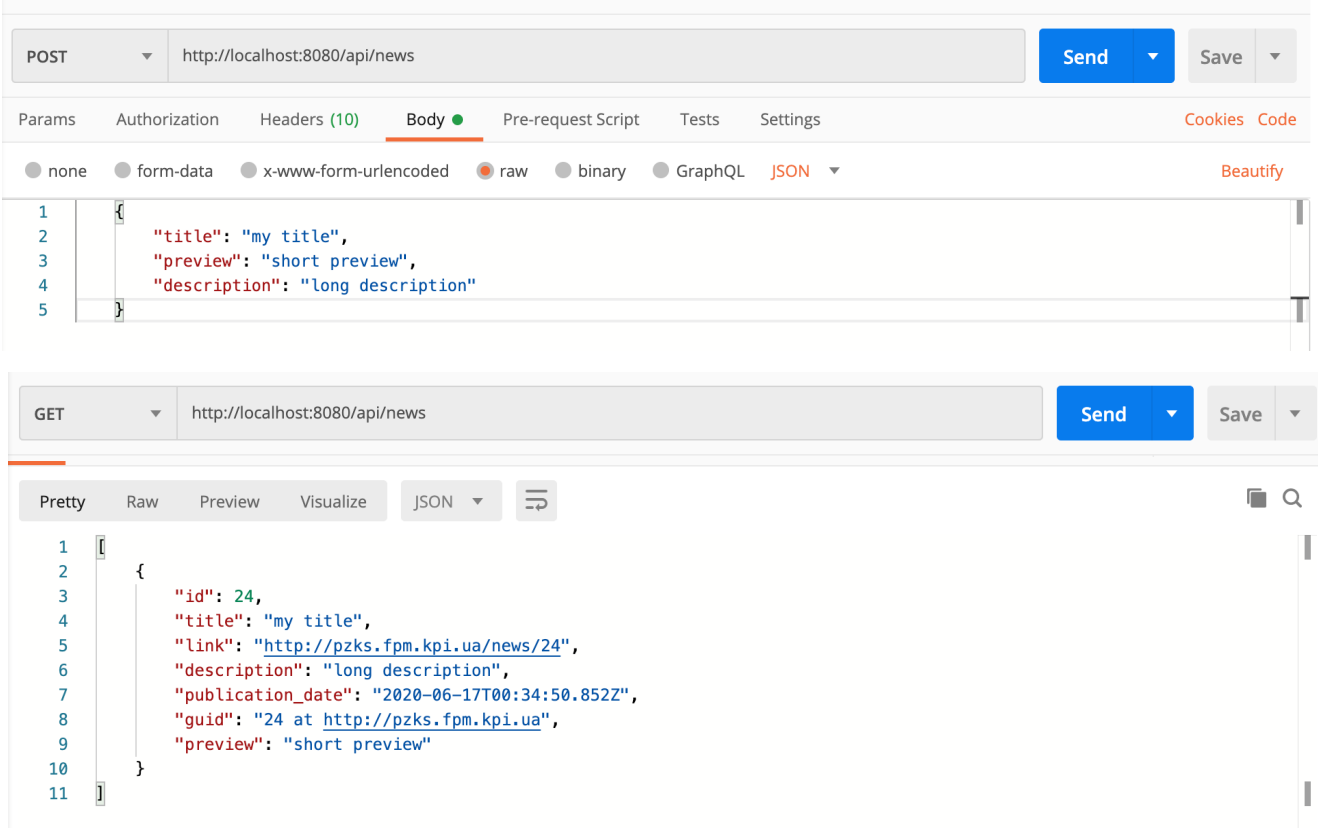
Застосувати зміни

СТРІЧКА НОВИН RSS

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
    <channel>
        <title>Кафедра програмного забезпечення комп'ютерних систем</title>
        <link>http://pzks.fpm.kpi.ua/</link>
        <description></description>
        <lastBuildDate>Tue, 16 Jun 2020 23:59:46 GMT</lastBuildDate>
        <docs>https://validator.w3.org/feed/docs/rss2.html</docs>
        <generator>https://github.com/jpmonette/feed</generator>
        <language>UK</language>
        <item>
            <title><![CDATA[Моя новина]]></title>
            <guid>23</guid>
            <description><![CDATA[Якась таблиця<br>
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>]]></description>
        </item>
    </channel>
</rss>
```

ТЕСТУВАННЯ ВЕБ-САЙТУ

Приклад тестування утилітою Postman:



РЕКОМЕНДАЦІЇ ДЛЯ ПОКРАЩЕННЯ

- впровадження системи захисту від DoS/DDoS-атак;
- реалізація шардингу та реплікаціїї бази даних;
- реалізація модулю статистики;
- створення архіву матеріалів;
- реалізація форми зворотного зв’язку;
- впровадження особистого кабінету користувача.

ВИСНОВКИ

1. Проведено збір вимог до системи
2. Створено архітектуру серверної частини системи
3. Створено структуру бази даних веб-сайту
4. Реалізовано модуль взаємодії з БД
5. Реалізовано модуль REST API
6. Налаштовано взаємодію серверної та клієнтської частин
7. Проведено тестування веб-сайту
8. Розроблено документацію проєкту

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2019 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-сайт кафедри програмного забезпечення комп’ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» розроблений згідно з архітектурним шаблоном SPA за допомогою мови JavaScript. Серверна частина реалізована за допомогою фреймворку Express.js.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Працездатність елементів сторінок веб-застосунку.
2. Коректність взаємодії з базою даних.
3. Стабільність роботи.
4. Забезпечення коректної обробки REST запитів.
5. Відповідність вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом White Box Testing. Таке тестування передбачає випадок, коли внутрішня структура застосунку відома. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування).
2. Тестування продуктивності програмного забезпечення, зокрема Performance testing (тестування стабільності).
3. Тестування граничних значень.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Для тестування застосовується утиліта Postman консольний інтерфейс взаємодії з БД psql та найбільш популярні веб-браузери.

Працездатність веб-додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля запитів.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування веб-сайту в різних веб-браузерах.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування правильності запитів до бази даних.
8. Тестування REST API.
9. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА
Керівництво адміністратора

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2020

ЗМІСТ

1. Процедура створення нової сторінки

Для створення нової сторінки потрібно відкрити проєкт у текстовому редакторі Після цього у дректорії «pzks_website/src/Project/» створити піддиректорію з назвою для майбутньої сторінки та скопіювати файл «pzks_website/src/Project/NewPageTemplate/index.js» у цю директорію. Далі потрібно змінити назву об’єктів виділених червоним (рис. 1) на назву новоствореної директорії. Далі всередині тегу «Semantic Container» потрібно створити розмітку нової сторінки. Для створення розмітки потрібно прибрати або модифікувати наявні елементи або додати нові в залежності від потреб. Для створення нових елементів потрібно використовувати HTML розмітку.

Після закінчення редагування файлу сторінки потрібно додати в інтерфейс посилання на неї. Для цього у файлі «pzks_website/src/Project/index.js» треба:

- додати імпорт нового об’єкту (рис. 2), де замість NewPageTemplate потрібно вставити назву новоствореного об’єкту а у лапках прописати шлях відносний шлях до файлу сторінки.
- додати новий тег Route, якому у якості атрибута path потрібно прописати частину URI яка буде використовуватися для посилання на сторінку, та замінити NewPageTemplate в атрибуті render на ім’я об’єкту (рис. 3).

Рис. 1. Створення об’єкту нової сторінки

Рис. 2. Імпорт об’єкту з іншого файлу

Рис. 3. Створення посилання на сторінку

Цих змін достатньо для того щоб після перекомпіляції проєкту нова сторінка була доступна за посиланням, що було вказано у атрибуті path, доданим до доменного імені веб-сайту.

Для того щоб додати посилання на вибрану сторінку до розділу меню потрібно зробити наступне:

1. Якщо пункт меню існує – перейти до наступного кроку, якщо ні – створити необхідний пункт меню (процедура створення пункту меню описана далі у цьому розділі).
2. знайти необхідний елемент меню у файлі «pzks_website/src/Project/Layout/Header/header.static.data.js» та додати інформацію про нове посилання до subMenuItems даного елементу (рис. 4), де:
   – title – текст посилання українською мовою;
   – titleEn – текст посилання англійською мовою;
   – path – посилання на сторінку без вказання доменного імені.

```
{
  id: 6,
  title: "Міжнародна діяльність",
  titleEn: "International Collaborations",
  subMenuItems: [
    {
      title: "Міжнародна діяльність",
      titleEn: "International Collaborations",
      path: "/international-collaborations",
    },
    {
      title: "Проект MEDIS",
      titleEn: "Project MEDIS",
      path: "/project-medis",
    },
    {
      title: "Проект PARIS",
      titleEn: "Project PARIS",
      path: "/project-paris",
    },
    {
      title: "Назва посилання українською",
      titleEn: "Liink title English",
      path: "/new-page",
    },
  ],
},
```

Рис. 4. Створення нового посилання у розділі меню

Для створення нового розділу меню потрібно у файлі «pzks_website/src/Project/Layout/Header/header.static.data.js» додати новий елемент за зразком (рис.5), де :

- id – ідентифікатор, поставити значення більше на 1 за значення у попередньому елементі;
- title – назва розділу українською мовою;
- titleEn – назва розділу англійською мовою;
- subMenuItems – посилання на сторінки, що належать до даного розділу меню

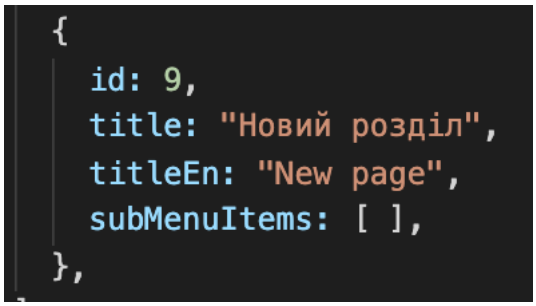


Рис. 5. Створення нового розділу меню

2. Процедура редагування контенту

Після проходження процедури авторизації, користувачу надається доступ до панелі адміністратора (рис. 6). Ця панель дозволяє редагувати присутню на сайті інформацію та інтерфейс веб-сайту. Дані представлені для редагування у форматі JSON. Принцип редагування однаковий для кожного розділу. Для запобігання загромадження інтерфейсу інформація у кожному розділі представлена у згорнутому вигляді. Для того щоб згорнути або розгорнути елемент використовують стрілки розміщені зліва від елементу (рис. 7).

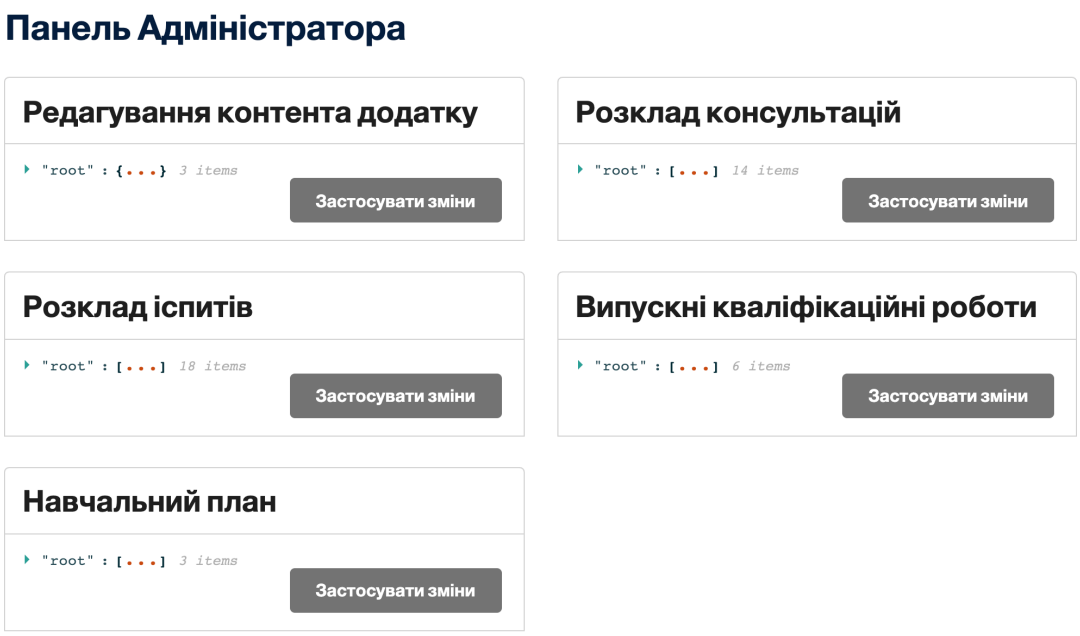


Рис. 6. Панель адміністратора

Рис. 7. Елементи згортання та розгортання блоків інформації

При наведенні курсору миші на поле об’єкту справа від нього з’являється дві кнопки (рис. 8). Перша з них (лівіша) дозволяє скопіювати значення поля до буферу обміну. Друга (правіша) дозволяє редагувати значення цього поля. Після натискання на кнопку редагування дані відображаються у текстовому полі і їх можна змінювати як звичайний текст.

Справа від поля з’являються дві кнопки (рис. 9):

- червоний хрестик – скасовує внесені зміни;
- зелена галка – підтверджує внесені зміни.

Рис. 8. Кнопки взаємодії з полем об’єкту

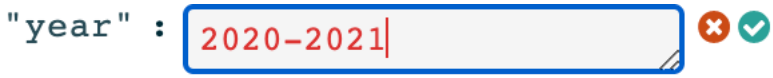


Рис. 9. Зміна значення поля

Після внесення змін кнопка «Застосувати зміни» стає активною, змінюючи свій колір з синього на сірий. Її натискання дозволяє зберегти внесені зміни у системі (рис. 10). Для застосування змін потрібно почекати деякий час, що приблизно дорівнює одній хвилині, доки проєкт перекомпілюється на сервері.

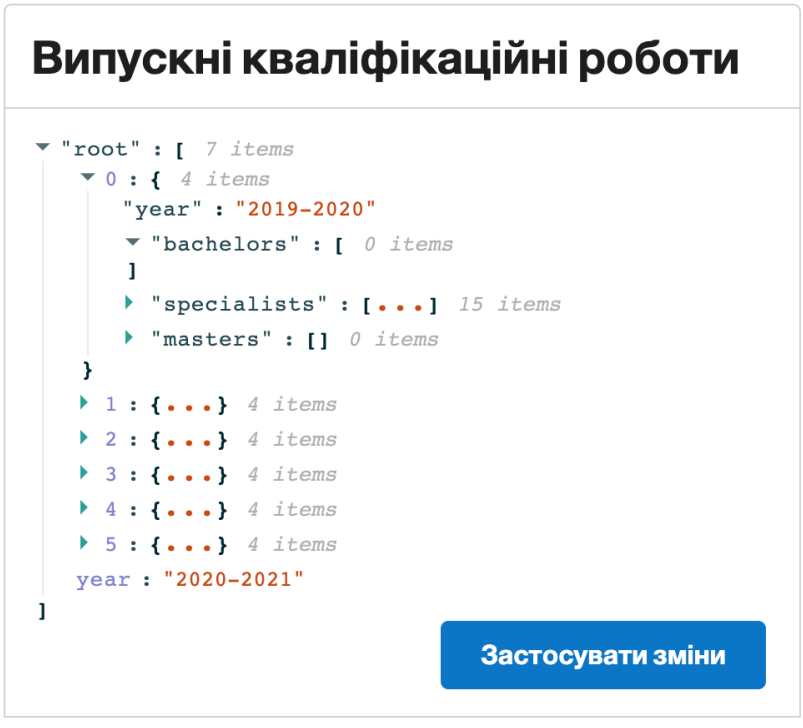


Рис. 10. Кнопка «Застосувати зміни»

3. Керування процесами сервера

Для керування процесами сервера використовується менеджер процесів pm2. Основні команди:

- pm2 list – виводить список процесів;
- pm2 start – запускає вказаний процес на виконання;
- pm2 stop – зупиняє вказаний процес;
- pm2 restart – перезапускає вказаний процес;
- pm2 delete – видаляє вказаний процес.

Використання цих команд напряму не є необхідним – для роботи з застосунком було створено окрему конфігурацію, що їх використовує (рис. 11).

```
"scripts": {
    "start:prod": "pm2 start npm --name \"server\" -- run start:server",
    "start:dev": "webpack-dev-server --hot --open --mode development",
    "start:server": "node server.js",
    "build": "rm -rf dist && webpack --config webpack.config.production.js ",
    "now-build": "npm run build",
    "test": "xo && stylelint './src/**/*.js'"
},
```

Рис. 11. Конфігурація скриптів для серверу

Для того щоб зміни на сервері були застосовані до веб-додатку потрібно:

- Якщо процес веб-додатку вже виконується, треба виконати bash script описаний у файлі «pzks_website/restart_server.sh»;
- Якщо процес веб-додатку ще не запущено, потрібно використовуючи командний рядок спочатку ввести *npm run build* для компіляції проєкту та *npm run start:prod* для запуску серверу.